



UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot•your knowledge partner



A Knowledge Approach to Software Testing

Essack Mohamed

Student Number: 13976419



**Thesis presented in partial fulfillment of the requirements for the degree of
Master of Information and Knowledge Management at the University of
Stellenbosch**

Mr D F Botha

Department of Information Science

December 2004

DECLARATION

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted at any university for a degree.

Signature : Date

ABSTRACT

The effort to achieve quality is the largest component of software cost. Software testing is costly - ranging from 50% to 80% of the cost of producing a first working version. It is resource intensive and an intensely time consuming activity in the overall Systems Development Life Cycle (SDLC) and hence could arguably be the most important phase of the process. Software testing is pervasive. It starts at the initiation of a product with non-execution type testing and continues to the retirement of the product life cycle beyond the post-implementation phase.

Software testing is the currency of quality delivery. To understand testing and to improve testing practice, it is essential to see the software testing process in its broadest terms – as the means by which people, methodology, tools, measurement and leadership are integrated to test a software product.

A knowledge approach recognises knowledge management (KM) enablers such as leadership, culture, technology and measurements that act in a dynamic relationship with KM processes, namely, creating, identifying, collecting, adapting, organizing, applying, and sharing. Enabling a knowledge approach is a worthy goal to encourage sharing, blending of experiences, discipline and expertise to achieve improvements in quality and adding value to the software testing process.

This research was developed to establish whether specific knowledge such as domain subject matter or business expertise, application or technical skills, software testing competency, and whether the interaction of the testing team influences the degree of quality in the delivery of the application under test, or if one is *the* dominant critical knowledge area within software testing. This research also set out to establish whether there are personal or situational factors that will predispose the test engineer to knowledge sharing, again, with the view of using these factors to increase the quality and success of the 'testing phase' of the SDLC.

KM, although relatively youthful, is entering its fourth generation with evidence of two paradigms emerging - that of mainstream thinking and that of the complex adaptive system theory. This research uses pertinent and relevant extracts from both paradigms appropriate to gain quality/success in software testing.

OPSOMMING

By verre die grootste komponent van sagte ware koste is dié verwant aan kwaliteitsversekering. Toetsing van sagte ware is koste intensief en verteenwoordig tussen 50% en 80% van die kostes om 'n beta weergawe vry te stel.

Die toetsing van sagte ware is nie alleenlik duursaam nie, maar ook arbeidintensief en 'n tydrowende aktiwiteit in die sagte ware ontwikkelings lewensiklus en kan derhalwe gereken word as die mees belangrike fase. Toetsing is deurdringend – dit begin by die inisiëring van 'n produk deur middel van nie-uitvoerende tipe toetsing en eindig by die voleinding van die produklewensiklus na die implementeringsfase.

Sagte ware toetsing word beskou as die geldwaarde van kwalitatiewe aflewering. Om toetsing ten volle te begryp en die toepassing daarvan te verbeter, is dit noodsaaklik om die toetsproses holisties te beskou – as die medium en mate waartoe mense, metodologie, tegnieke, meting en leierskap integreer om 'n sagte ware produk te toets.

'n Benadering gekenmerk deur kennis erken die dinamiese verhouding waarbinne bestuurselemente van kundigheid, soos leierskap, kultuur, tegnologie en maatstawwe reageer en korrespondeer met prosesse van kundigheid, naamlik skep, identifiseer, versamel, aanpas, organiseer, toepas en meedeel. Die fasilitering van 'n benadering gekenmerk deur kennis is 'n waardige doelwit om meedeling, vermenging van ervarings, dissipline en kundigheid aan te moedig ten einde kwaliteit te verbeter en waarde toe te voeg tot die proses van sagte ware toetsing.

Die doel van hierdie navorsing is om te bepaal of die kennis van 'n spesifieke onderwerp, besigheidskundigheid, tegniese vaardighede of die toepassing daarvan, kundigheid van sagte ware toetsing, en/of die interaksie van die toetsspan die mate van kwaliteit beïnvloed, of een van voorgenoemde die dominante kritieke area van kennis is binne die konteks van sagte ware toetsing. Die navorsing beoog ook om te bepaal of daar persoonlike of situasiegebonde faktore bestaan wat die toetstegnikus vooropstel om kennis te deel, weer

eens, met die oog om deur middel van hierdie faktore kwaliteit te verbeter en die toetsfase binne die sagte ware ontwikkelingsiklus suksesvol af te lewer.

Ten spyte van die relatiewe jeudigheid van die bestuur van kennis, betree dit die vierde generasie waaruit twee denkwyses na vore kom – dié van hoofstroom denke en dié van ingewikkelde aangepaste stelselsdenke. Hierdie navorsing illustreer belangrike en toepaslike insette van beide denkwyses wat geskik is vir meedeling van kennis en vir die bereiking van verbeterde kwaliteit / sukses in sagte ware toetsing.

ACKNOWLEDGEMENTS

I dedicate this research to my wife and first born, who endured with me, exercised great patience and without whose love, strength and encouragement I would not have completed.

To my brother, Abubaker who I silently admire as being an exemplar of principles, my gratitude.

I am grateful and indebted to a number of individuals including Professor Ben Fouche for his wisdom and knowledge transfer; Dr Martin van der Walt for his guidance and openness. Mr Daan Botha for his frankness and direction, Marijke Weitsz for her support and encouragement, Lesinda Pretorius for her generosity, unselfish friendship, her constancy and for always being there – she continues to guide and be a source of inspiration. Nick Fallon for his sincerity, unceasing concern, his reliance and integrity.

My sincere thanks to Mariana Cooper for taking on the typeset – making a tedious task seem so simple.

And thanks to two very giving individuals, Miss Louise Tucker and Carmen Davids who were invaluable resource investigators for providing much research material including a treasure of web sites and contact details.

To George Botha for his continued support and guidance – a special and supportive friend.

I am indebted to each of these individuals – to all of them my warmest thanks.

TABLE OF CONTENTS

	<u>Page</u>
CHAPTER 1	11
INTRODUCTION	11
LEADERSHIP IN SOFTWARE TESTING	11
SOFTWARE TESTING - a BRIEF HISTORY	12
CRITICAL KNOWLEDGE AREAS FOR SOFTWARE TESTING	13
OBJECTIVES OF THE RESEARCH	14
CHAPTER 2	15
METHODOLOGY	15
INTRODUCTION	15
THE HYPOTHESES	17
THE STRUCTURE AND FLOW OF THIS PAPER	18
CHAPTER 3	19
A KNOWLEDGE CONTEXT	19
INTRODUCTION	19
MAINSTREAM VERSUS COMPLEX THEORY	21
THE IMPORTANCE OF KNOWLEDGE MANAGEMENT	22
KNOWLEDGE TAXONOMIES	24
KNOWLEDGE SHARING and ORGANISATIONAL CULTURE	27
ESSENTIAL KNOWLEDGE IN SOFTWARE TESTING	30
SUMMARY	32
CHAPTER 4	33
SOFTWARE TESTING	33
INTRODUCTION	33
SOFTWARE QUALITY IN PERSPECTIVE	34
SOFTWARE TESTING CONTEXT and SKILLS	35
MODELS, TEST TECHNIQUES, and LIFE CYCLES	35
A MODEL SOFTWARE TEST SYSTEM	38
SOFTWARE TESTING PROCESSES	39
SOFTWARE TESTING CYCLES	39
TEST METHODOLOGY	40
TESTING PHASES AND TEST TYPES	41
TEST COVERAGE	42
TESTWARE	44
TEST CASES (TEST CASE LIFE CYCLE)	44
BUG CYCLE	45
SUMMARY	45
CHAPTER 5	47
DOMAIN KNOWLEDGE, APPLICATION/TECHNICAL EXPERTISE & MATURITY LEVELS	47
INTRODUCTION	47
DOMAIN KNOWLEDGE	48
APPLICATION / TECHNICAL KNOWLEDGE	49
GENERAL PROFESSIONALISM	50
MATURITY LEVELS in SOFTWARE TESTING	52
SUMMARY	58
CHAPTER 6	59
DATA ANALYSIS	59
INTRODUCTION	59
DATA GATHERED	60
DEMOGRAPHIC DATA	61
TYPE OF EMPLOYMENT	61
COMPANY SIZE	61
AGE AND GENDER	62
EDUCATION AND TRAINING	63
WORK EXPERIENCE	63
ATTITUDE TO KNOWLEDGE SHARING	64

BENEFITS AND DRAWBACKS OF KNOWLEDGE SHARING	65
BENEFITS	65
PERCEIVED DRAWBACKS	66
FACTORS INFLUENCING SOFTWARE QUALITY	66
OWN KNOWLEDGE VERSUS ACCESS TO KNOWLEDGE	67
DOCUMENTS PRODUCED BY TEST ENGINEERS.....	69
AREAS TO COLLECT AND SHARE DATA WITHIN SOFTWARE TESTING DISCIPLINE	70
TEST MATURITY LEVELS.....	70
CHAPTER 7	73
FINDINGS AND CONCLUDING THOUGHTS	73
INTRODUCTION	73
PERSONAL FACTORS INFLUENCING KNOWLEDGE SHARING	74
EXTENT OF KNOWLEDGE SHARING.....	74
ENABLING - ATTITUDE AND WILLINGNESS	74
KNOWLEDGE STOCK.....	75
RISK REDUCTION.....	75
ORGANISATIONAL CULTURE	76
BARRIERS TO KNOWLEDGE SHARING	76
BENEFITS OF KNOWLEDGE SHARING.....	76
FACTORS INFLUENCING QUALITY OF DELIVERY IN SOFTWARE TESTING	77
DOMAIN KNOWLEDGE.....	77
SOFTWARE TESTING COMPETENCY	77
APPLICATION OR TECHNICAL EXPERTISE	78
EDUCATION AND TRAINING	78
KNOWLEDGE OF THE SDLC AND VARIOUS MODELS	79
IN SUMMARY, A KNOWLEDGE APPROACH TO SOFTWARE TESTING:	79
CONCLUDING THOUGHTS	80
TEXT REFERENCE NOTES	82
REFERENCES.....	83
APPENDIX A	91
SOFTWARE MODELS	91
APPENDIX B	94
BUG CYCLE	94
APPENDIX C	96
EMPLOYEE BENEFITS - AN OVERVIEW AND ARCHITECTURAL COMPONENTS.....	96
EMPLOYEE BENEFITS IN LAY TERMS	96
BENEFIT STRUCTURE	97
EMPLOYEE BENEFITS ARCHITECTURAL COMPONENTS	99
SUMMARY.....	100
APPENDIX D	101
COVERING LETTER	101
RESEARCH SURVEY: A KNOWLEDGE APPROACH TO SOFTWARE TESTING IN AN	
EMPLOYEE BENEFITS ENVIRONMENT.....	101
APPENDIX E.....	103
SURVEY INSTRUMENT.....	103
A KNOWLEDGE APPROACH TO SOFTWARE TESTING IN AN EMPLOYEE BENEFITS	
ENVIRONMENT	103
APPENDIX F.....	112
EARLY TESTING IS COST EFFECTIVE TESTING	112
APPENDIX G	114
GLOSSARY OF TERMS	114

LIST OF FIGURES

	<u>Page</u>
Figure 1: Organisational Knowledge conversion processes (Source: Nonaka & Takeuchi)	25
Figure 2: Skills continuum (Source: Rex Black, 2002)	31
Figure 3: Quality Assurance Components (Source: Lewis, 2000)	35
Figure 4: Waterfall methodology (Source: Lewis, 2000)	36
Figure 5: V- Model	37
Figure 6: Test System (Source: Black, 2002)	39
Figure 7: Test Cycle (Source: Test Methodology)	41
Figure 8: Test Spectrum (Source: Black, 2002)	43
Figure 9: A typical Test Case Life Cycle (Source: Black, 2002)	44
Figure 10: Test Maturity Model (Source: Burnstein et al, 2000)	53
Figure 11: Sample population – Type of Employment	61
Figure 12: Gender distribution	62
Figure 13: Categories of tertiary and Software Training.	63
Figure 14: Appendix A – Traditional Waterfall SDLC Methodology	91
Figure 15: Appendix A – Deming's PDCA quality circle	92
Figure 16: Appendix A – Spiral Development / Test Methodology	93
Figure 17: Appendix B – Bug Cycle (Source: Black, 2002)	94
Figure 18: Appendix C – EB Architectural Components (Source: Schmanan, 2003)	99

LIST OF TABLES

	<u>Page</u>
Table 1: Knowledge paradigms: Mainstream versus Complex Theory (Source: Stacey, 2001)	21
Table 2: Test Maturity Model adopted for a typical domain area (Source: Burnstein et al, 2000)	57
Table 3: Company Size	61
Table 4: Age statistics	62
Table 5: Work Experience	63
Table 6: Attitudinal response profile	64
Table 7: Benefits of Knowledge sharing	65
Table 8: Perceived Drawbacks of Knowledge sharing	66
Table 9: Factors affecting quality of delivery	67
Table 10: Own versus Access to knowledge	68
Table 11: Documents completed by Test engineers	69
Table 12: Areas of value to collect and share within Software Testing	70
Table 13: Test Maturity level in organisations	71
Table 14: Perceived Failure-Success rate	72
Table 15: Appendix C - EB Benefits and variables	98
Table 16: Appendix F - Early testing: Cost of error removal (Source: Paul; Automated Software Testing).	112

CHAPTER 1

INTRODUCTION

LEADERSHIP IN SOFTWARE TESTING

‘Improvements in quality always and automatically result in reductions in schedule and costs, increases in productivity, increases in market share, and consequently increase in profits.’

.....- .W M Deming

Software quality is something everyone wants. Today’s software managers are pressured constantly to bring quality products into the market with ever-shrinking schedules at minimal costs. Getting a product to market as early as possible may mean product survival or product death – and therefore company survival or death. In an attempt to do more with less, organisations want to test their software adequately, but within a minimum or optimal schedule (Paul et al, 1999).

To accomplish this goal, organisations are ever aware of the elements in the delivery chain that prospectively contribute to a shortened cycle without compromising the quality of the delivery. Amongst these are elements that distinguish leaders from followers at various levels of maturity, such as:

- The organisation’s Software Testing competency (Burnstein, Suwannasart & Carlson, 2000; Black, 2002; Kaner, 1999; Bach et al, 2002; Dustin, 2002; Beizer, 1984; Lewis, 2000)
- Domain knowledge (Black, 2000; Kit, 1995)
- Application knowledge (Black, 2000; Dustin, 2002; Kit, 1995)
- The organisation’s approach to knowledge and knowledge sharing (Snowden, 2003, Stacey et al, 2001, Sveiby, 1997; Stewart, 1997; Davenport & Marchand, 2002; Nonaka et al, 2000)
- Leadership skills, and leadership’s knowledge paradigm (Senge, 1990).

In order to maintain a leadership position in the commercial world, intelligent organisations leverage knowledge internally to survive externally (Stewart, 1997). Knowledge models (Arthur Andersen Model, Knowledge Management Reference Model, American Productivity and Quality Model, and others) have evolved over the years with emphasis on different enablers such as leadership, culture/structure, processes, technology and measures which act in a dynamic relationship with the so called KM processes, namely, create, identify, collect, adapt, organize, apply, and share. Although many espouse to be experts, KM is an emerging discipline that stresses a formalized, integrated approach to manage the complexity of relationships and knowledge resident with employees in a manner that will benefit the individual and the organisation simultaneously. This research, aware of KM enablers and processes, will pay greater attention to knowledge sharing and the application of knowledge to improve quality or gain success in software testing.

Companies are in agreement that a contribution to their competitive advantage is their 'brainware' or their 'human capital' (Stewart, 1997). It is not surprising to find the amount of knowledge that often-underrated individuals such as software testers (test engineers) accumulate during their tenure in a particular sector or in a specific functional area in an organisation.

SOFTWARE TESTING - a BRIEF HISTORY

From the literature it is clear that the history of software testing mirrors the evolution of software development itself (Paul, et al, 1999). For a long time, software development focused on large-scale scientific and military programs coupled with corporate database systems developed on the mainframe or minicomputer. Software testing and test scenarios during this era were written down on paper, and tests targeted control flow paths, computations of complex algorithms, and data manipulation. A finite set of test procedures could effectively test a complete system. Testing was generally not initiated until the very end of the project schedule and performed by personnel who were available at the time (Paul et al, 1999).

Technology, development and software testing has graduated and made great strides since then. The growth of the computer industry and introduction of personal computers, micro-electronics, and telecommunications gave birth to a new era and led to ubiquitous technology and commercial software development (Castells, 2000). Commercial software applications (e.g. a complex *Enterprise Resource Planning (ERP) Employee Benefits (EB)* 'package') compete for supremacy, speed to market and survival. A noticeable change is evident where such complex integrated systems increasingly move toward societal demands for greater member control and wider investment choice.

Customers require of companies with which they do business to continuously improve, particularly in speed of service, price, convenience and personalization (Kalakota & Robinson, 2000). This means the application architectures are becoming inherently more complex and it concomitantly demands that the test effort increase in rigor, that organisations are aware of 'best'¹ software test practices, apply appropriate test models and introduce formal structured test methodologies to ensure a test capability that will achieve quality defect free software to be supportive of an endless number of growing permutations and combinations. Only when the test process has been documented and metrics have been defined, collected and analysed (*a vital and intrinsic component of KM*), can the test team make effective improvements (Black, 2000).

CRITICAL KNOWLEDGE AREAS FOR SOFTWARE TESTING

Testing is connected to a lot of different activities and groups. Some critical processes are internal to the test team, carried out and affecting testers only, whilst other test processes are synergistic and require collaboration across teams. Each critical testing process is, to a greater or lesser extent, subject to the context in which it occurs. That is, the test process must fit the people, the system, the project, and the organisation involved.

The literature introduces four categories of skills to software test capability: general qualifications, software testing skills, domain knowledge, and application or technical expertise. This document touches on each of these critical areas of knowledge with the aim

of determining the affect on the quality of the application (known as the Application Under Test (AUT)) or software delivery.

OBJECTIVES OF THE RESEARCH

This research will explore a knowledge approach to software testing by looking broadly at knowledge, KM, culture, and the critical knowledge areas in software testing. The primary objective will also include the extent to which knowledge is being shared in software testing, the factors that may predispose individuals to knowledge sharing with a specific focus on software testing, general professionalism, domain knowledge and application/technical expertise.

As a secondary objective we look at a maturity grading of knowledge in software testing through a hierarchical process. For this the research looked at the Test Maturity Model (TMM) initiated by the Illinois Institute of Technology (Burnstein et al). The TMM contains a set of maturity levels through which an organisation can take incremental steps to progress to greater software testing grade.

In brief, the objectives of this research are:

- To establish the importance of software testing knowledge in delivering quality
- To establish domain knowledge and it's influence on the quality of effective software testing.
- To establish application/technical knowledge and it's influence on the quality of effective software testing.
- To assess if demographics (personal or situational) predisposes test engineers to sharing knowledge.
- To investigate the extent of knowledge sharing in the discipline of software testing.
- To determine if one area of software testing knowledge is more dominant or more critical relative to any other.

Finally, findings and conclusions will be presented with the intention of contributing to and showing how a knowledge approach advances the objectives of software testing.

CHAPTER 2

METHODOLOGY

INTRODUCTION

This research will make use of both primary and secondary sources² to assess current practices and trends regarding knowledge sharing in software testing, and to determine the influence of domain and application/technical knowledge on effective software testing. Whilst Employee Benefits (EB) is used as an example of an application domain area, the survey and interviews extend to other domain areas as a pragmatic juxtaposition. The findings can be applied to areas wider than EB.

Literature and functional areas of information about current software testing practices and KM are the main sources for this paper, while the extent of knowledge sharing and test engineers' knowledge predisposition were collected, analyzed and interpreted using semi-structured interviews and survey questionnaires.

There is an abundance of literature on the subject that provide for either a situational paradigm or a conceptual knowledge framework. However, the literature does not provide a clear coherent progression of applying these concepts into organisational action. Despite the popular and relatively mature Capability Maturity Model, the literature is not clear in providing an assimilating framework that combines capability, software testing maturity and the sharing of knowledge within the organisation.

This research is designed to include the obvious and documented aspects of software testing and use of knowledge, and the linkage of these to affect a quality delivery in a functional domain. Information obtained from primary and secondary sources include questionnaires, interviews, journals, Internet searches and academic writings and books written on software testing, systems development life cycles, effective quality delivery, KM, knowledge

processes, enabling knowledge, learning, as well as meetings with senior managers in organisation's that shared their experiences and their documentation (predominantly in an EB domain area). Interviews conducted with test tool vendors and test service suppliers proved useful and complementary.

With respect to software testing maturity and knowledge approaches the Institute of Software Engineering's (ISE) Capability Maturity Model (CMM) and Test Maturity Model (TMM) is referenced as an objective basis to assess how organisations may graduate from their current software testing position.

To define meaningful definitions, content and views on software testing, enabling knowledge, domain knowledge and application/technical expertise, and general professionalism the research drew from some of the noted authorities in the specific fields. For example, in the Software testing field from Beizer, Black, Dustin, Kit, Kaner, Graham, Perry & Rice, De Marco & Lister, Lewis, Schach, Laudon & Laudon and from the KM field Snowden, Stacey, Penrose, Senge, Prahalad & Hamel, Nonaka & Takeuchi, Davenport, Prusak, Sveiby, Zack, and Quality Assurance field Crosby, Deming, Rommel, Swieringa & Wierdsma to name but a few.

The views and opinions of these and other authors are incorporated in the paper to sketch a framework, to contribute to an improved understanding of software testing, and to facilitate a knowledge approach to software testing.

By way of primary source of data, semi-structured interviews were conducted and questionnaires were emailed to a select group of professionals and practitioners. The main purpose of the interviews and informal discussions was to assess the test engineers' predisposition to knowledge sharing, to determine the extent of knowledge sharing in software testing and also to provide substantial qualitative data.

The interviews facilitated definition, and contributed to refining the scope of the research. To eliminate bias and to increase the randomness and breadth of the survey, questionnaires that were sent to the select group were forwarded by these candidates to an additional group

of participants within their areas of influence. Respondents used the offered email address to communicate and send their responses.

No respondent was required to divulge any private or confidential information as the data focuses on experience rather than on a specific organisation or project (Refer to Appendix D – Covering Letter).

The findings from the interviews and questionnaires were analysed using Microsoft excel (Refer to Appendix E – Survey Instrument).

THE HYPOTHESES

H0: There is no relationship between demographic factors (age, education, training, experience, and organisation size) and the test engineer's predisposition to knowledge sharing and its application.

H1: There are select demographic factors that predispose a test engineer to knowledge sharing and knowledge application.

H0: Insignificant knowledge sharing and knowledge application is currently evidenced in the software testing discipline.

H2. Knowledge sharing and its application are widely used in the software testing discipline.

H0 There is no relationship between domain knowledge and the impact on software testing.

H3 The test engineer's level of domain knowledge has a direct and proportional influence on software testing.

H0 There is no relationship between application/technical knowledge and (its impact on) software testing.

H4 The extent and depth of the test engineer's experience of the application itself has a marked positive influence on software testing.

THE STRUCTURE AND FLOW OF THIS PAPER

- Chapter 3 provides a brief contextual KM framework.
- Chapter 4 presents a macro view of software testing and demonstrates the importance of software testing competency in delivering quality.
- Chapter 5 provides a view of the influence and importance of domain knowledge and application/technical expertise to effective software testing. This chapter concludes with progressive maturity levels of a knowledge paradigm in software testing using the Test Maturity Model (TMM).
- Chapter 6 presents an overview of the data collected from interviews and the structured questionnaire, and makes reference to certain knowledge components (used by the survey instrument) that the literature draws attention to.
- Chapter 7 concludes the research with a final discussion of the findings.

CHAPTER 3

A KNOWLEDGE CONTEXT

INTRODUCTION

This research is about a knowledge approach to Software Testing. The approach is aggregated from two sources; firstly data gathered through the medium of a survey instrument and interviews, and secondly through the medium of published literature. The literature presents an evolving view of knowledge and KM. Since the early 1990s KM evolved from being primarily techno-centric to a second-generation resource based stage followed by a third generation focused on content and taxonomies. The current emerging generation of KM has its emphasis on the complexity of humans and the complex responsive processes in organisations, in particular complex adaptive systems (CAS).

Dr Michael Koenig, dean of the College of Information and Computer Science (Long Island University), says that KM has gone through three distinct phases already in its short lifetime. The first technology stage includes information technology, intellectual capital and the Internet. The second stage is the human resources stage that is premised on the fact that technology is not good enough if people are not motivated to use it. The subsequent stage is the content and taxonomies stage ensuring that knowledge is made accessible and usable - where mainstream knowledge and KM traditionally focuses on the conversion of tacit to explicit knowledge.

We now evidence a fourth generation of KM emerging rapidly as an alternative to and challenging the traditional mainstream thinking, challenging some fundamental premises on which the previous KM generations were built.

It must be emphasised that whilst these are 'ostensibly' alternate views with some authors very strongly expressive away from the mainstream, the central and common theme throughout the evolutionary stages of KM is the improvement of decision-making and the

creation of conditions for innovation. This research explores Software Testing in an inclusive knowledge context – extracting from each knowledge generation the salient, appropriate and critical aspects to software testing that addresses largely the risk areas (quality, time, costs and resources).

It is a truism that KM is not going away. Supporting this statement is the research done by International Data Corp (IDC) who state that poorly managed knowledge costs Fortune 500 organisations about \$12 billion a year due to substandard performance, intellectual rework and a lack of available KM resources (Swartz, 2003).

The literature presents certain basic truths that precede KM as a specific discipline.

These are:

1. That the knowledge, ideas and expertise of a company's employees are indeed valuable assets.
2. Knowledge can only be volunteered; it can never be conscripted.
3. We only know what we know when we need to know it.
4. We always know more than we can say, and we always say more than we can write down.
5. It is advantageous for companies to encourage their employees to share their know-how and expertise with the rest of the staff so that the rest of the organisation can benefit (albeit in a formal, informal or semi-informal basis). This concept is being broadly spoken of in the current KM CAS generation and coined by Snowden as just-in-time (JIT) KM.

Whether we approach Software Testing using the traditional mainstream KM thinking or the emerging theory of complexity, or a hybrid thereof, there are a number of practical challenges that confront organisations embarking on knowledge programmes. Not least of which is the creation and sharing of knowledge, changing personal knowledge disabling or predisposed attitudes, building a knowledge enabling organisation and overcoming organisational barriers. This subset of identified challenges are pertinent and used in this research to elicit responses in the survey instrument and interviews.

Interviews and discussions with various organisations showed that it is not uncommon to find that the application domain knowledge, the detailed gems are ‘islands of knowledge’ housed in the minds of those that have been in an environment and with the organisation for more than 10 or 15 years.

This chapter presents knowledge in context of Software Testing by starting off introducing the pertinent differences between mainstream knowledge thinkers and complex advance systems theorists. This is followed by broadly describing the importance of knowledge, context and definition as presented by authors reviewed in the literature, moving on to briefly discussing an enabling knowledge culture and knowledge sharing, and finally presenting the critical knowledge areas in software testing.

MAINSTREAM VERSUS COMPLEX THEORY

This section draws out the salient differences between mainstream knowledge thinkers and complex responsive process perspective with respect to software testing.

	Mainstream	Complex Adaptive Systems
Process of relating between people	<u>Individual's</u> mind as mental model, knowledge inside his or her head	Experience in both social and individual minds.
New knowledge creation	Primarily in the <u>individual</u>	No distinction between individual and organisational knowledge
Knowledge transfer	Explicit knowledge is that knowledge which an individual is aware of and can articulate. Tacit Knowledge is transmitted by stories that members tell each other (Communities Of Practices).	Knowledge is transferred by the ongoing participation in patterns of inter-relationships

Table 1: Knowledge paradigms: Mainstream versus Complex Theory (Source: Stacey, 2001)

Being aware of the challenges and differences between mainstream knowledge thinkers and CAS – this research will look at areas that are both common (relevant) and important to software testing.

THE IMPORTANCE OF KNOWLEDGE MANAGEMENT

In knowledge organisations there is less machinery and more employees.

‘The basic economic resource – the means of production – is no longer capital, nor natural resources, nor labour. It is and will be knowledge’ (Drucker, 1993).

Mainstream knowledge thinkers present the view that knowledge is primarily information within people’s minds; without a knowing, self-aware person there is no knowledge. Knowledge is highly valuable, because humans create new ideas, insights and interpretations and apply these directly to information and decision-making (Davenport, Marchand, 2000).

The alternative CAS theory espouses the view that the individual and the social (or teams in the organisation) is one level. In context of effective software testing, we embrace a duality wherein knowledge of both the individual (mainstream) and the team (CAS theory) are critical to support the quality of delivery whether it be structured formally, informally or semi-informally. The key is interaction, managing the relationships and co-operation for sustaining ongoing knowledge sharing and learning in a competitive environment.

From the alternative CAS process perspective individual minds are continuously reproduced and knowledge assets lie in the pattern of relationships between its members (Stacey, 2001). KM, if conducted professionally, is considered an asset that can derive continuous and sustainable economic value for an organisation over its lifetime (Stewart, 1997).

This affirmation is intuitively known and recognised, particularly by businesses that have recently experienced a depletion of knowledge in key areas. It is also acknowledged as a high risk, virtually a threat, by organisations that have years of domain knowledge treasured and vested in one or two individuals only. Businesses that begin KM initiatives recognise the importance of employees’ knowledge and that this knowledge is a valuable resource too. The

financial effects of losing productivity, knowledge and experience coupled with tedious resource selection, recruitment and training cycles to replace lost knowledge can be staggering.

In many cases the actual impact, or the gravity of its value is not readily realised until the knowledge that has walked out the front door is needed (Hansen & Thompson, 2002).

One of the challenges facing KM is what knowledge to manage and to what end? KM activities are all over the map: building databases, establishing corporate libraries, building intranets, leading cultural change, fostering collaboration, creating virtual teams, stimulating interaction – all of these are shades of KM and each is approached differently depending on perspective.

The myriad of definitions that attempt to describe knowledge make it inherently difficult to manage, audit, plan or control knowledge. Below is a selection of definitions from many available, from experts in the field of KM:

- Laurence Prusak (IBM): 'KM is the attempt to recognise what is essentially a human asset buried in the minds of individuals, and leverage it into an organisational asset that can be accessed and used by a broader set of individuals on whose decision the firm depends.'
- J.I. Hansen, C.A. Thompson (2001) 'Knowledge Management at Work' offers a definition that focuses primarily on resources: 'KM is nothing more or less than the deliberate management of three resources – *people, process and technology* – to put the intellectual capital of a company to work.'
- J. Fenn (Lead Analyst, Gartner) states that KM is a discipline that promotes an integrated approach to identifying, managing and sharing all of an enterprise's information assets.
- Sveiby defines knowledge as the capacity to act.
- Skyrme defines knowledge as information with meaning.
- Perhaps the most pragmatic categorisation is cited by Charles Savage (Skyrme, 1999):

- Know-how - a skill, procedures
- Know-who - who can help with this question or task
- Know-what - structural knowledge, patterns
- Know-why - a deeper kind of knowledge; understanding the wider context
- Know-when - a sense of timing, and rhythm
- Know-where - a sense of place, where it is best to do something
- Stacey defines knowledge as the evolutionary process of reproduction and potential transformation at the same time. Organisational knowledge is not located anywhere but arises continually in the relationships between people within organisations.

Knowledge constitutes value and worthy assets in the company. These assets are the properties of complex adaptive systems that underpin value creation and future earnings potential. Managing the organisation's knowledge effectively and exploiting it internally and in the marketplace is the latest pursuit in seeking competitive advantage (or sustaining the strategic vision in certain organisations).

A common aspect of both mainstream and CAS theory is for business to translate information into knowledge, and to establish a sharing culture to ensure analysis, interpretation, responsible communication and effective decision-making within the context of business strategy.

Before embarking on knowledge sharing and culture, a discussion of knowledge taxonomies is important to illustrate where mainstream KM and CAS theory or JIT KM fundamentally deviate and where different perceptions or 'types' of knowledge have different management implications.

KNOWLEDGE TAXONOMIES

Mainstream knowledge thinkers such as Nonaka & Takeuchi (1995) identify two types of knowledge: tacit and explicit.

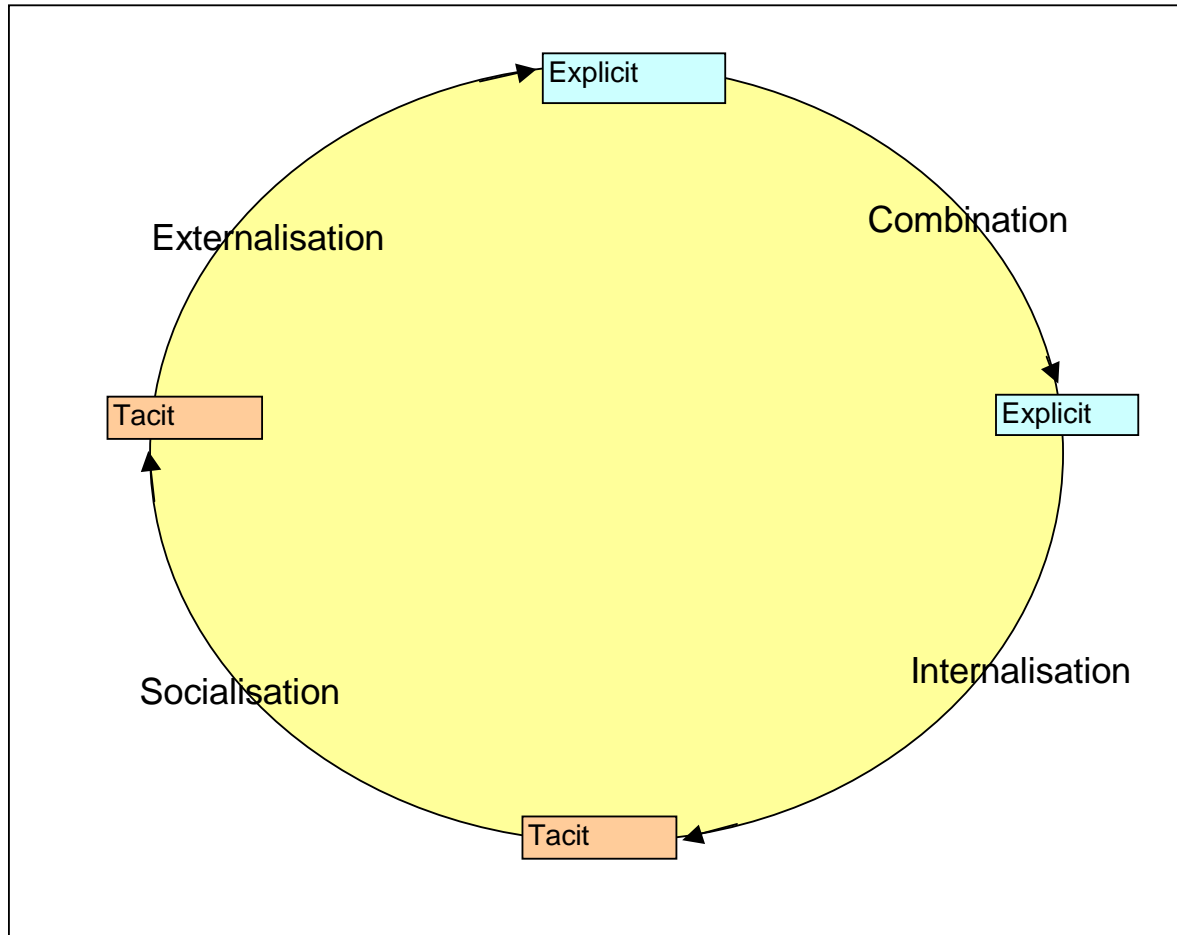


Figure 1: Organisational Knowledge conversion processes (Source: Nonaka & Takeuchi)

According to Nonaka and Takeuchi (1995), the fundamental reason why Japanese enterprises have become successful is because of their skills and expertise at organisational knowledge creation. Knowledge creation is achieved through the recognition of synergistic relationship between tacit and explicit knowledge in the organisation, and through the design of social processes that create new knowledge by converting tacit knowledge into explicit knowledge.

The mainstream thinking posits that tacit knowledge is personal knowledge that is hard to formalise or communicate to others. It consists of subjective know-how, insights, and intuitions that have come to a person from having been immersed in an activity for an

extended period of time (e.g. the domain knowledge found in individual's heads referred to earlier). Explicit knowledge on the other hand is formal and tangible knowledge that is easier to transmit between individual groups.

These categories of knowledge are complementary. Tacit knowledge *while it remains closely held as personal know-how*, is of limited value to the organisation. Explicit knowledge does not appear spontaneously, but must be nurtured and cultivated.

Organisations must create a supportive culture that encourages openness and sharing of knowledge. Figure 1 shows four modes of knowledge conversion: from tacit knowledge to tacit knowledge through a process of socialisation, from tacit knowledge to explicit knowledge through externalization, from explicit knowledge to explicit knowledge through combination, and from explicit knowledge to tacit knowledge through internalization. The four modes feed off each other in a continuous spiral of organisational knowledge. Knowledge creation typically begins with individuals who develop some insight into how to do their tasks better (e.g. creating a central corporate software testing database and getting members to contribute to it). As long as knowledge stays tacit, the organisation is unable to exploit it further. Organisations need to become skilled at converting personal tacit knowledge into explicit knowledge (Choo, 1998).

In summary it is important to motivate access to knowledge through improved information management, procedures, process management and structured methodologies, that is, facilitating sharing of knowledge becomes an important part of KM.

The alternative complex responsive process perspective holds the view that tacit and explicit knowledge are facets of the same communicative process and, therefore, it makes no sense to talk about them separately or to believe that one is converted into the other.

CAS theory maintains that communicative interaction is a human relationship that is a living process, which cannot be captured, stored or owned by anyone. People participate in relationships that are mutually constructed and none of them can individually own their process of mutual construction (Stacey, 2001).

What is clear from the literature is that both the mainstream and CAS theory have important

common grounds and in agreement - each framework endorses stimulating the creation of organisational structures where trust and common context naturally occur with a view to gain knowledge sharing and improve decision making.

KNOWLEDGE SHARING and ORGANISATIONAL CULTURE

Unlike most other assets, knowledge alone has minimal or no value. *In reality, it's not what you know that gives power; it's what you share about what you know that gives you power* (Hansen & Thompson, 2002).

Mainstream knowledge thinkers espouse the view that knowledge resident in the minds of individuals needs to be converted into organisational knowledge that can be shared.

(Choo;1998). Knowledge unlike other assets, does not suffer from diminishing returns.

Instead, as knowledge is shared, its value goes up (Hansen & Thompson, 2002).

It is possible to describe many everyday activities – from reading Test Strategy, Test matrices, Test templates or a business requirements document to chatting with a colleague at the water cooler – as knowledge oriented. However, such activities, valuable as they may be, cannot easily be used to develop concrete measures of the prevalence of knowledge in organisational processes.

Spontaneous, unstructured knowledge transfer is vital to a firm's success (Davenport & Prusak, 1998). Knowledge is shared in the organisation whether it is specifically managed or not (refer to survey – this aspect is briefly discussed in the analysis chapter). When an employee asks a colleague how to perform a task he is requesting a knowledge transfer (Davenport & Prusak, 1998). Successful transfer implies that the receiving party accumulates new knowledge.

On a subconscious level, individuals may not realise the value of what they know (hidden or passive knowledge), and they may not know how to communicate what they know effectively to the right person at the right time. This requires creating an enabling knowledge culture in the organisation. In the 'knowledge-enabled' organisation, workers routinely capture, document, and share knowledge. The cultivation and incentive is entrenched in the dynamic

of the business infrastructure and importance is placed in human resources for the good of the individual and for the greater good of the organisation.

The complex responsive process view of organisational knowledge creation and knowledge sharing leads to different conclusions to those perspectives proposed by mainstream thinkers. The CAS theory is common in its thought that knowledge is found in the actions of relations between people. The difference between the two being that CAS theory appears to project that it is the *only* manner in which knowledge themes are experienced in organisations.

With respect to this research its attention is on knowledge sharing and the organisational culture stimulating this dimension.

Therefore in agreement of both the mainstream and CAS perspective, the important question is: 'does the organisation's culture reward decisions and actions according to how people use and share their knowledge?'

The entire gambit of testing in a complex domain environment involves planning, analyzing, designing, developing, executing, reporting and other business activities that depend heavily on knowledge sharing. If knowledge workers feel that they have no time to share knowledge in their course of work, or that it is inconvenient to do so, even the best models or repositories will be of no use.

Management involvement is key. The lack of management projects a lack of perceived need and may lead to inadequate allocation of time and resources to the initiative. Lacking management involvement, the culture has little hope of changing to support knowledge sharing, and the concept of KM becomes nothing more than a passing fad. Management and key resources can fall victim to entrenched ways of thinking that inhibit knowledge sharing. They may feel that admitting what they do not know as something that will threaten their position of power, or by sharing their knowledge diminishes their power base (this concept is tested in the survey).

A number of individual behaviours and attitudes may be ingrained and inherently discourage knowledge sharing. Although learning and development is at its best when reviewing

mistakes, there is an inevitable reluctance to discuss foregone errors. (Hansen & Thompson, 2002).

Project failures are not easily shared. In this light poor test planning at the start of a project that result in poor test execution, may only be detected after implementation - this often gets ignored.

Further, cultural or organisational factors that inhibit or discourage sharing, as identified by Hansen & Thompson are:

- The 'Silo'³ company is unwilling to share internally fueled by lack of leadership, incentive, and support within the organisation
- The 'not invented here' syndrome where employees are unwilling to look at existing knowledge (whether elsewhere inside the company or outside the company).
- No standards and procedures – each department uses its own vocabulary and methods to process or store knowledge.

CAS theory proposes that you cannot get everyone to believe the same thing, but you can institute rituals that align people (Snowden, 2002).

Interviews with key members of certain organisations introduced this research to the awareness of the organisation's empowering policy with respect to knowledge sharing. However, in the wake of changes in the general commercial environment and specifically changes to organisational structure (positions being repurposed) together with the looming prospect of retrenchment, management acknowledge that a negative atmosphere exists, which is contrary to a knowledge-enabling or sharing culture.

The difficult task is to change, actually to motivate policies, and practices that encourage the cultivation and proper usage of knowledge assets. This may entail changes to core functions such as hiring, reviews, performance appraisals, performance contracts, reward systems and promotion practices.

Knowledge sharing is an essential ingredient for success in testing especially in a complex domain environment. It is a commodity that must be pursued and harnessed as an invaluable nugget. Testers need to be knowledgeable to be effective and efficient (Black, 2002).

ESSENTIAL KNOWLEDGE IN SOFTWARE TESTING

Since knowledge is rooted in human experience, competence and social context, managing it well means paying attention to people, organisational culture, technology and their inter-relationships.

The capabilities of the testing team can greatly affect the success, or failure, of the testing effort. An effective testing team includes a mixture of technical and domain expertise relevant to the software problem at hand (Dustin, 2003). It is not enough for a testing team to be equipped with the testing techniques and tools only. Depending on the complexity of the domain, a test team should also include members who have a detailed understanding of the problem domain. This knowledge enables them to create effective test components and data and to implement test cases, test scripts and other test mechanisms effectively (Dustin, 2003).

From the literature it is clear that a first step is to find out what the essential knowledge areas are that constitute a competent test team. Figure 2 shows four critical categories of skills identified by Black (2002):

- General qualifications
- Testing skills
- Domain skills
- Application/ technical expertise

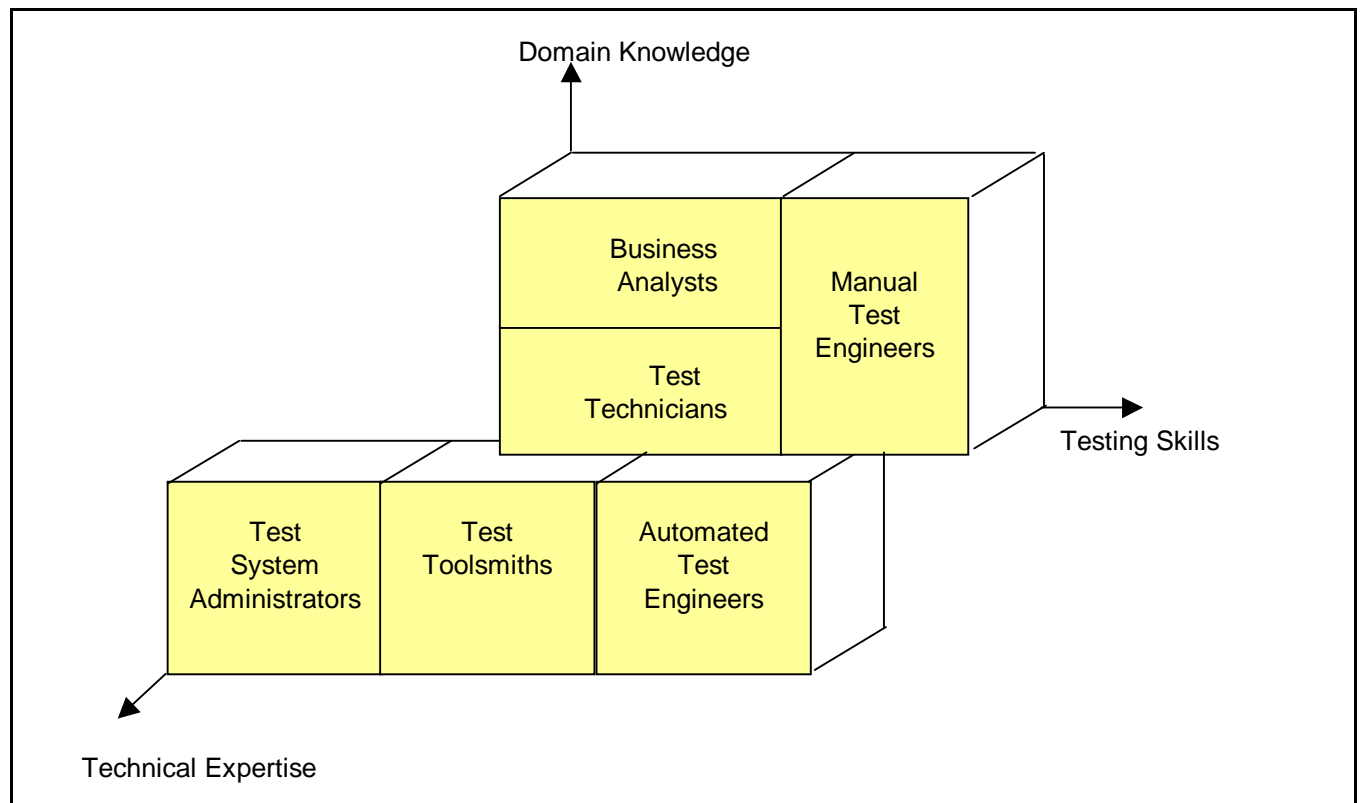


Figure 2: Skills continuum (Source: Rex Black, 2002)

The first category, general qualifications, relates to tertiary education or a post-matriculation qualification when recruiting a test engineer. What is important is the seeking of the discipline of a similar nature or basic skill level within the individual to hold the position.

The literature warns that a successful, cost-effective contingent is one that is applicable to a particular project as well as one that will be most useful to the test team. That is, no one solution will fit all situations (Paul et al, 1999).

The chapters that follow focus on three knowledge areas:

- testing skills (chapter 4),
- domain knowledge (chapter 5) and
- application/technical expertise (chapter 5).

SUMMARY

- Knowledge in mainstream thinking shifts the paradigm and places the recognition of a knowledge approach clearly centred on a person, and thereby putting emphasis on converting tacit individual knowledge into explicit for organisational benefit. From a complex responsive process perspective, individual minds and the more repetitive social experience arise together in the interaction between people. What makes action more effective is the quality of the contribution into the ongoing flow of relationships.
- It is not exactly known what knowledge exists within a person's brain, and whether he or she chooses to share knowledge.
- Organisations must foster and encourage a knowledge sharing environment by motivational leadership, enabling culture and providing the appropriate infrastructure, methods and processes.
- If for no other purpose but to the extent that Software Testing in its promise to deliver defect free software, is premised mainly on achieving predictable results, testers and knowledge workers of organisations are encouraged to share their knowledge.
- General qualifications, testing skills, domain knowledge and application/technical expertise are critical knowledge areas to software testing.

CHAPTER 4

SOFTWARE TESTING

INTRODUCTION

'Software Testing is fundamental to delivering quality software on time within budget (Ed Kit, 1995). Software Testing is a paradox. If we could test just the parts of the system that we suspect of having defects, we could drastically reduce the amount of time spent on testing; but defects are usually found in places we least suspect – so we dare not assume that any area of the system is defect-free (Perry & Rice, 1997).

The basic goal for software quality is that it performs its functions in the manner that was intended by its architects. In order to achieve this goal, the final product must contain a minimum of mistakes in implementing their intentions (Deutsch, 1982). In essence software testing is not Quality Assurance (QA), but a key part of QA.

The success of a delivery and possibly the success of the organisation, albeit a complex delivery or a discrete functional component, may rest on the quality, depth, deployment and management of software testing (Kit, 1995).

Failure in the market or losing market share is the kind of bad news that may result directly from poor software testing or from the inability to manage 'constructive failure' or from a significant derivative or product of software testing.

In the foregone years technology and application domain expertise were well recognized and highly remunerated whereas testing skills were under appreciated. Testers were thought of as subservient and sourced from failed developers. This has changed – software and applications are no longer islands of data or just interfacing information - it has become powerfully integrated across projects, across organisations and, with the internet, it has become global. Software testing has earned a respectable position in the SDLC. Software

Testing has entered a new era as opposed to the days when virtually one single individual could test a complete system effectively by a finite set of test procedures.

The literature positions testing as a science that is increasingly becoming knowledge oriented, and with the increasing growth in technology, software testing is becoming a specialist field. It is important to have an independent testing function with a viable position in the organisation, and it is becoming necessary to hire technically competent people and reward them adequately. Testing is to be seen as a profession with its own work products that are maintained as important assets of the organisation (Kit, 1995).

This chapter's objective is:

- to present Software Testing in the context of QA.
- to provide an overview of Software Testing with the aim of highlighting the critical knowledge areas.
- to extract Software Testing 'best'¹ practices from the literature describing professional practitioner's activities.
- to show the importance of software testing knowledge to achieve quality delivery in the SDLC.

This research emphasises the knowledge areas critical to software testing by presenting a high-level overview of software testing. It is not the intention to discuss in any detail any aspects of software testing areas or its sub processes.

SOFTWARE QUALITY IN PERSPECTIVE

As stated earlier in chapter 1, software quality is something everyone wants. For example, managers know that they want high quality; software developers know they want to produce a quality product; and users insist that software work consistently and reliably (Lewis, 2000).

Successful managers that deliver consistent quality know how to make people quality conscious and make them recognize the benefits of quality. Broadly speaking QA can be divided into 3 parts (Figure 3), Software Testing, Configuration Management, and Quality Control.

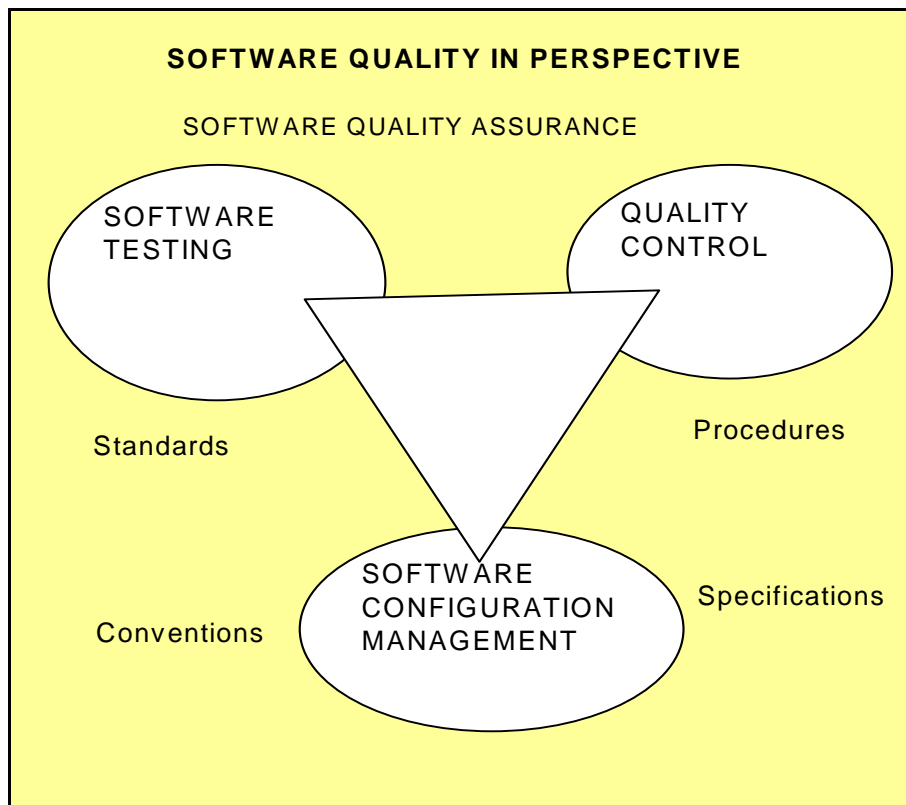


Figure 3: Quality Assurance Components (Source: Lewis, 2000)

The success of QA depends not only on the careful management of the parts, it also depends on a coherent collection of standards, practices, conventions, and proper specifications. The lack of proper standards and procedures is a significant disabler to knowledge sharing and, if left unattended, will disadvantage the prospective knowledge developing organisational culture.

SOFTWARE TESTING CONTEXT and SKILLS

MODELS, TEST TECHNIQUES, and LIFE CYCLES

To get software error free is a tremendously onerous and responsible task. The quality of the software is dependent on a number of factors, some of which are the particular test model employed for the situation, the test system, the level of maturity, and the knowledge and competency of resources in the test team. The two latter factors are dealt with in a subsequent chapter.

We must recognize that software testing is pervasive⁴ and as a result, it influences the test models and test methods used in organisations by individuals that interpret, analyse and apply it to the situation at hand, or apply it according to experience or knowledge disposition (Burnstein et al, 2000).

There are several process models shown in Appendix A. Each model requires a good knowledge of testing and the correct context within the SDLC in order to select the most appropriate for the project or AUT.

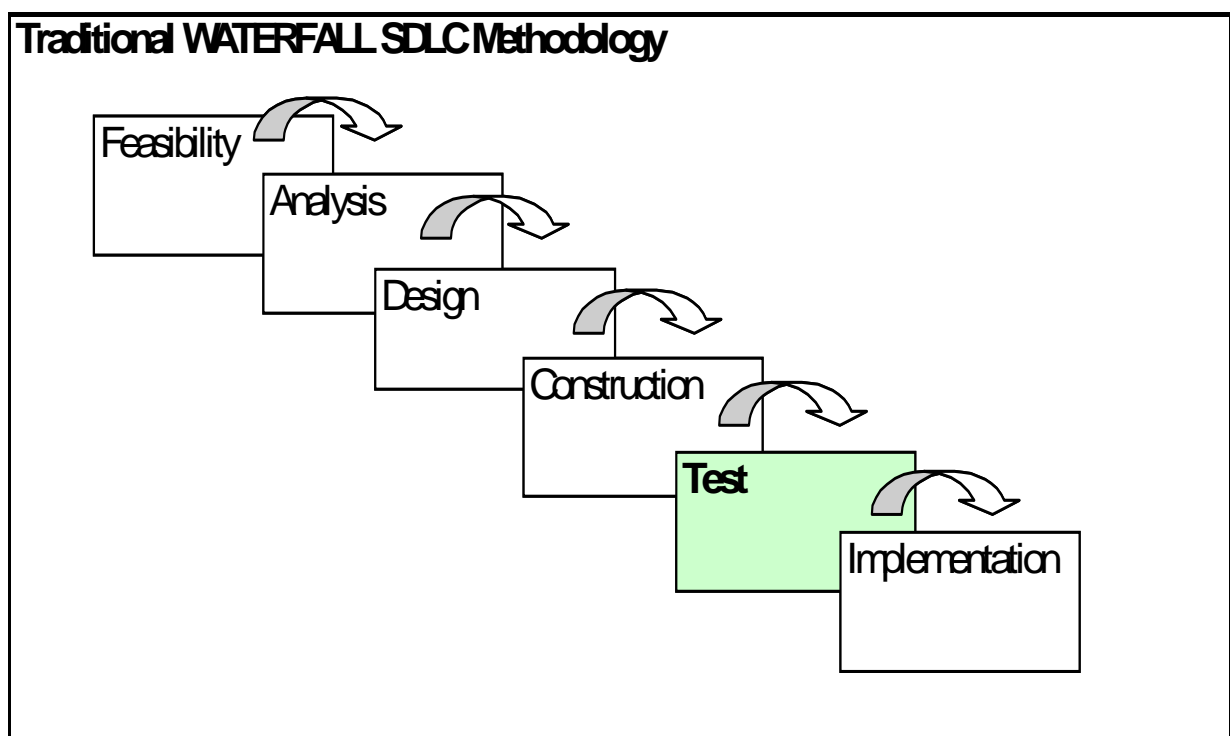


Figure 4: Waterfall methodology (Source: Lewis, 2000)

Pervasive is best demonstrated by the more popular V-model (Figure 5).

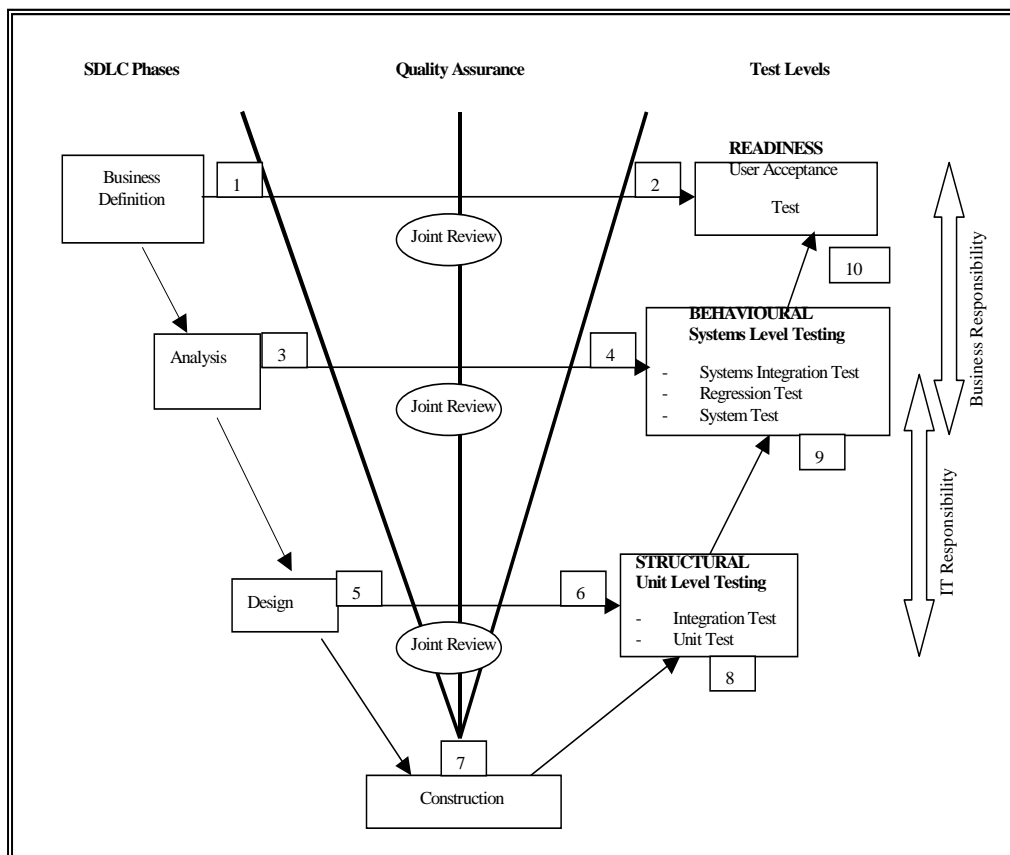


Figure..... The V Model

Figure 5: V- Model

The literature suggests that Software testing starts early in the SDLC. The V-model shows that testing occurs in parallel with the development cycle and is a continuous process. It is explicit in starting the test process early, as opposed to the traditional validation testing which suggests that testing starts after the coding phase has been completed. Testing should be integrated into application development or in the case of an acquired package, concurrently with the configuration task.

Being equipped with test models is insufficient. Black quotes George Box that states 'all models are wrong; some models are useful'. Each lifecycle model is designed to deal with certain kinds of project risks more effectively. To be optimal or to be able to choose the most appropriate one requires both SDLC and test knowledge. When initiating a delivery the question should be asked whether another model, and therefore another test strategy, would not fit better in the context of the delivery (Black, 2002).

Choosing the correct model affects the management, design and deployment of a congruent test system. Designing and implementing an effective and efficient test system will allow the correct assessment of the quality of the systems under test.

Before embarking on a comprehensive test system the literature advises to scan the knowledge base, solicit knowledge assets from previous projects, gain knowledge from domain subject matter experts to assist in determining what *may/may NOT* be tested (scope) against what *should* be tested (user view) and compare this to what *can* be tested (budget and resource constraints) (Black, 2002). The next section shows the components that are essential to a good test system and what is central to it. The model illustrates a test system and must not be read as the test system – test systems will vary from project to project.

A MODEL SOFTWARE TEST SYSTEM

A test system consists of three major elements as shown in Figure 6. The ‘test system’ is the organisational capability for testing created by:

- testing processes
- the testware, and
- the test environment.

Depending on the complexity of the AUT, a test system plan, design and implementation may begin months before the AUT is delivered to the test team. Organisations differ in their creation and the storage medium of a test system – sometimes it exists on personal computers on personal databases, or files. There are organisations where test systems exist entirely in the heads of knowledge workers.

Figure 6 illustrates the core elements of a software test system.

An interdependent test system helps the test team focus testing efforts on key quality risks, and to find, reproduce, isolate, describe and manage the most important bugs in the AUT.

Central to the test system is a *competent test team* to ensure the consistent provisioning of effective and efficient services (Black, 2002).

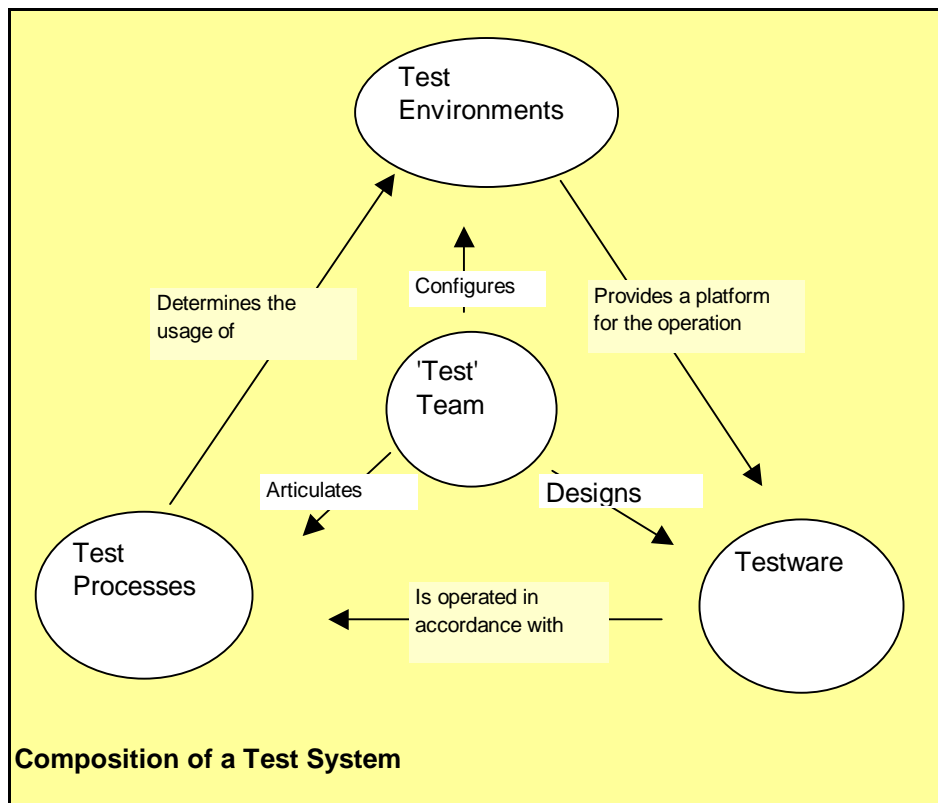


Figure 6: Test System (Source: Black, 2002)

The optimal test system is knowledge based that requires a combination of skills and experience associated with each of the three major elements shown in Figure 6.

SOFTWARE TESTING PROCESSES

SOFTWARE TESTING CYCLES

Since the software development life cycle is an iterative process, software testing follows in tandem with this life cycle approach. Life cycle testing means that testing occurs in parallel with the development life cycle and is a continuous process (as described using the V-model). After the initial release of a product, any change to the product should require that development and testing activities revert to the life cycle phase that corresponds to the type of change made. For example, if a new function is added to the application (not instigated by a change in requirements), then a new functional design specification is required, and the process should revert to the functional design phase and continue sequentially from there (Kit, 1995). In other words all development and testing activities do not get lumped into the

operation/maintenance phase in an ad-hoc or informal manner as a result of the application or product being released to users/ customers.

A structured knowledgeable approach is expected in a complex domain environment, and because testing spans the entire SDLC (V-model) employing a methodology makes project sense and makes software testing better to manage.

TEST METHODOLOGY

A mature more formal approach is to adopt a test methodology that is cognisant of application size and complexity, the test environment and the competency of resources that are available to deliver effectively across the spectrum of the test cycle.

A methodology covers every aspect of software testing. It should be platform independent and span the entire SDLC as shown in Figure 7.

As a minimum a test methodology will include the following Key Process Areas (KPA's):

- test planning
- test case development
- test environment preparation
- test execution
- test results analysis and
- management reporting activities.

Each of the KPA's requires specific knowledge and specific test skills. The literature recommends that different roles within the test team assume responsibility for a particular KPA.

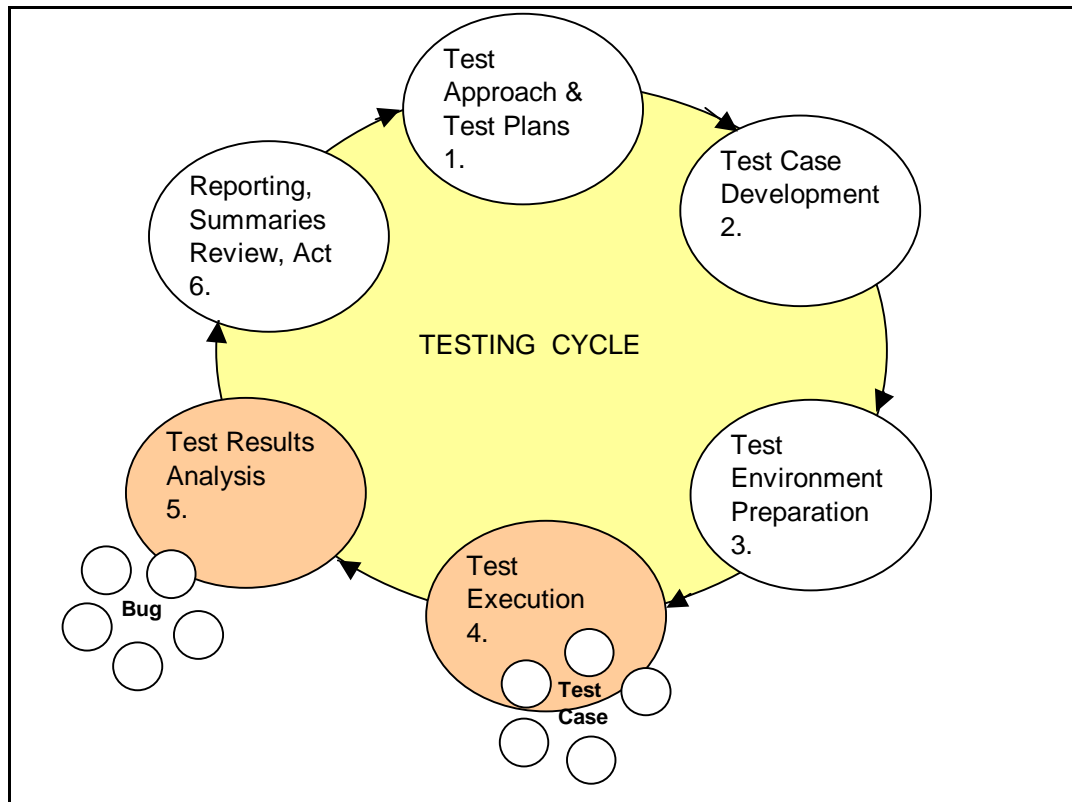


Figure 7: Test Cycle (Source: Test Methodology)

Planning is crucial – it is the roadmap to test strategy and provides guidance and direction to the test team. One activity of the test plan is breaking the system under test into distinct test types within phases.

TESTING PHASES AND TEST TYPES

In an unstructured and immature environment, the period of test execution activity during development or maintenance is sometimes an undifferentiated blob. In such an environment testing begins, testers may run some vaguely defined tests and identify some bugs, and then at some point the project manager declares the testing complete (Black, 2002).

As the organisation matures it gains more experience and collects more knowledge. In this manner a more disciplined approach of partitioning testing into a sequence of phases is adopted. The various phases differ from organisation to organisation, from application to

application, from project to project – they may even have different classifications, names or hold different meanings within the organisation (Silo effect mentioned in chapter 3).

Sequencing specific parts of the testing process to fit into the larger SDLC in terms of timing is a challenge. This will be addressed according to the AUT, the organisational structure, the resource constituency and competency, the complexity of the delivery and, not least, the number of concurrent deliveries.

A test plan would advocate rigorous test phases to accomplish effective granularity and tractability from business requirements through to technical specifications, and plan for the appropriate test type(s) within each phase.

A key consideration in the test plan is to know when enough tests have been created. There are various ways in which the completeness of testing referred to in the literature as test coverage can be measured.

TEST COVERAGE

Whilst aware of the number of techniques, it is also acknowledged that testing is a discipline and requires tight focus. It is easy to try to do too much. For example, besides the models described, there are an infinite number of complementary techniques, test scenarios and test cases that one *could* analyze, design and execute for an application to be accurately tested. Clearly for a complex domain application, the test effort would be excessively time-consuming and expensive.

Even if you try to focus on what you might think to be ‘good enough’ quality, you can find that such testing is too expensive or that you will experience trouble finding out what ‘good enough’ means for colleagues, managers, users or clients (Black, 2002).

As easy as it is to try to do too much, it is equally important to ensure that sufficient testing is done to mitigate any risks of defects slipping through the test cycle.

Test coverage is best viewed in terms of granularity. Test granularity refers to the fineness or coarseness of a test’s focus. Test granularity can be thought of as running along a spectrum (right hand-side of V-model) ranging from structural to readiness as shown in Figure 8.

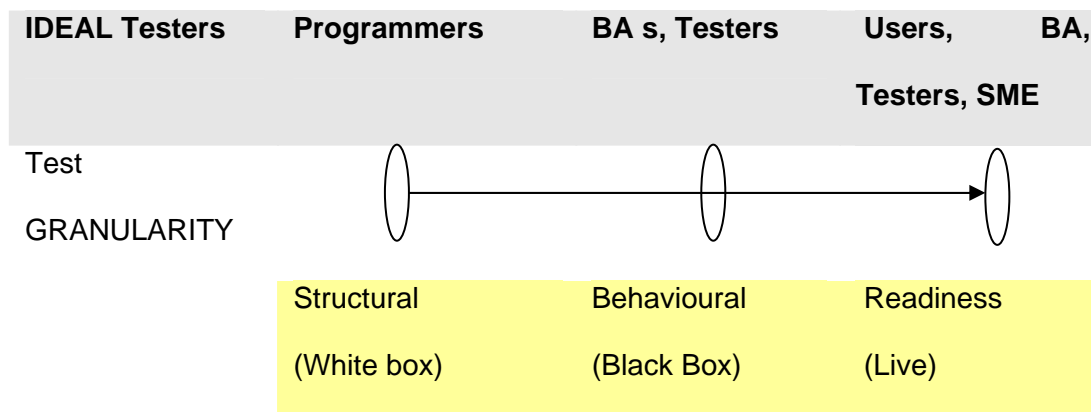


Figure 8: Test Spectrum (Source: Black, 2002)

- Structural (white box) tests refer to finding bugs at the lower level operations. These structural tests are based on how a system operates. To test effectively in this area, knowledge of the technical specification and the programming language is essential
- Behavioural (Black-box) tests refer to finding bugs in the high level operations. These tests involve a detailed understanding of the application domain. To test effectively in this area, knowledge of the functional specification and the architectural designs is essential
- Readiness tests include users, content or subject matter experts, and early adopters. To test effectively in this area, knowledge of the business requirements and processes is essential

Interviews showed that although much of the content is essential to effective testing and that testing is dependent on all three stages of granularity, it is still found in people's heads rather than in formal documents. The lack of standards and documentation inhibits the potential corporate knowledge stock and thereby reduces the chances of future value to the organisation.

To facilitate, manage and control the testing cycle, testing produces its own formal documents corresponding to the life cycle, and these form part of testware.

TESTWARE

Testware includes test plans, templates, test cases, test data, test scripts, test tools, test logs, test reports, checklists, supporting documentation like specifications, test procedures and associated user guides or documentation. Two of these are touched on in this section.

TEST CASES (TEST CASE LIFE CYCLE)

On any given project across the test types within a test phase the most important aspect is the development and tracking of test cases. As stated earlier testing is about risk management, and risk cannot be managed without superb test cases or without good test data. Like the hapless predator at the end of the food-chain, picking up every toxin that trickles through the ecosystem, testing in many organisations - picks up every undetected failure, delay or slip at the tail-end of the development that can manifest itself into confusion and missed dates (Black, 2002).

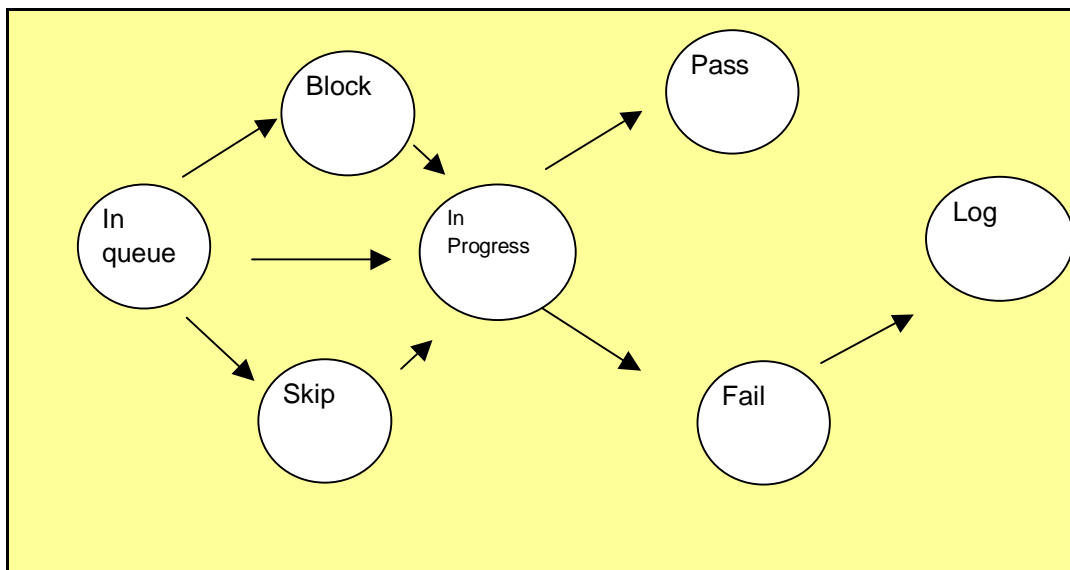


Figure 9: A typical Test Case Life Cycle (Source: Black, 2002)

The test team needs a knowledge base and mechanism to track, analyse, and present what's going on in the test domain and to ensure coverage is achieved.

Good test cases well managed through the means of a robust test-case life cycle, elicits useful information about the quality of the system under test as shown in Figure 9. The test

case cycle 'ends' at logging a problem, which in turn initiates the next cycle, the bug (defect) cycle.

BUG CYCLE

A bug is understood to be a problem present in the system under test that would cause it to fail or failed to meet an expected condition. Another way of describing it is 'a bug is a potential source of product dissatisfaction.'

Bug management, documenting, tracking and resolution is a specialist knowledge niche in testing – it takes significant effort and discipline, and it provides some important benefits, for example:

- Facilitates clear communication about the quality of the system under test.
- Useful audit device as evidence of execution.
- Good trend analysis – with predefined metrics.
- Provides severity analysis and therefore priority resolution.
- Keeps attention focused.
- Provides new information as bugs age.
- Every bug detected is one less that could have been shipped.

The literature recommends that a bug report go through an identifiable cycle with clear ownership at each phase. A typical bug life cycle is illustrated in Appendix B.

SUMMARY

- Software testing is a significant part of QA.
- The success of QA depends on a coherent collection of standards, practices, conventions, and proper specifications. The lack of proper standards and procedures or the inconsistency thereof is a significant disabler to knowledge sharing.
- Software testing touches many areas and many activities in the SDLC. It is pervasive. Early and timeous test involvement in the life cycle yields best results for the AUT.

- There are several life cycle models; George Box says 'all models are wrong; some models are useful'. Each lifecycle model is designed to deal with certain kinds of project risks more effectively. To be optimal or to choose the correct model one needs knowledge or access to experience.
- Critical knowledge areas of software testing consist of three major elements:
 - test processes, testware and test environments with a knowledgeable test team at the fulcrum.

Knowledge of test planning, test cycles, the key process areas, test phases, test types, test error management and competency in the area of coverage sufficiency form part of the these areas

- From the literature it is clear that one may consider the possibility of accomplishing a delivery of software without adequate testing expertise, but it certainly would not have the mark of quality.
- There is a growing perception that software quality is a weak link in developing high quality products. The ability to develop and deliver reliable bug free software continues to elude many organisations. It is knowledge of testing, the knowledge of models, of adopting good planning, preparation, execution and measurement that contribute to increasing quality for the AUT. A lot of this knowledge is the domain of individuals and not owned by the organisation. The literature advises that mechanisms be put in place to encourage knowledge sharing, and to document knowledge throughout the test cycle in a central knowledge database for later reference and value to projects still to come.

CHAPTER 5

DOMAIN KNOWLEDGE, APPLICATION/TECHNICAL EXPERTISE & MATURITY LEVELS

INTRODUCTION

From interviews and the literature it is clear that the position you put people into depends to some extent on their level of experience. Software testing jobs are no different and will generally require someone who has specific slant of experience. While some authors prefer seeing more experience in testing when choosing members to join a test team – despite recognizing that domain knowledge, application or technical expertise be more important – there is agreement in the literature that it depends on the needs of the project and the current strengths and weaknesses of the existing team.

From the literature it is clear that software testing competency, domain knowledge, application/technical expertise and overall professionalism are variables that markedly influence the quality of delivery. Depending on the nature of the application, for example, being in charge of a nuclear device, the test manager better be assured that every member on the test team be top-notch, experienced and deeply knowledgeable about the system under test. In contrast, a service software product (*such as an ERP EB package*) a test manager may accept or tolerate a few resultant interpretation errors (Black, 2002).

It can be said that the more scientific the application the greater the degree of building accurate test cases and documenting expected results. Whereas the more service-centric or consumer-centric the application, the greater the degree of inherent uncertainty and volatility will be. This is as a result of the ever-changing needs of customers and of service providers to match the constantly evolving customer needs (see Appendix C). This latter situation will evidence more tolerance from a test manager and cater for a qualified margin of error.

In the first chapter we identified elements that distinguish leaders from followers in software testing at various levels of maturity. We also discussed a knowledge context and software testing in previous chapters. The objectives for this chapter are:

- To discuss domain knowledge and its influence on the quality of effective software testing.
- To discuss application/technical knowledge and its influence on the quality of effective software testing.
- To briefly discuss the general professionalism required of testers and how it may influence the quality of effective software testing.
- To position software testing in a knowledge graded maturity hierarchy.

DOMAIN KNOWLEDGE

The requirements for testing skills, application knowledge or technical expertise and domain knowledge differ from project to project (Black, 2002). Domain knowledge in a word processor for example, where the majority of expert users (writers, authors, scripters, lecturers and so forth) use about 20% of all the features, is it expected that the test team have a significant level of domain knowledge? On the contrary if the team is expected to test geological software for oil exploration, and the in-house organisation's competency is not sufficient, the test team will invite external expertise to assist in the delivery of quality (Black, 2002).

The literature also provides an alternative position with a caveat worth discussing. In summary it states that the less the test team or a tester knows about the application domain, the less likelihood of bias there is in the tests and the more likely that the test will reveal incorrect functions, missing functions, and improper assumptions. However, this would mean (as discussed in Chapter 4 under the subsection describing test coverage) that more tests, test cases and test conditions would be unproductive and inefficient (Beizer, 1984).

Therefore, whilst software managers are constantly pressured into bringing quality products into the market with ever shrinking schedules at minimal costs, the dual positions debating

the issue whether tests are either incomplete or too time consuming means a compromise between thoroughness and budget. On the one hand the detailed domain knowledge allows the elimination of countless test cases that would prove nothing thereby gaining huge timesavings. Whilst on the other hand the 'independent' tester would more likely find more missing functions, is inherently inefficient because he cannot take advantage of domain knowledge but perhaps gains more on quality.

Another scenario where domain knowledge comes to bear is when requirements to create a test system (see chapter 4) are inadequately documented. The importance of having testable, complete and detailed requirements is always desired, but in practice having a perfect set of requirements at the test team's disposal is a rarity. In absence of proper requirements or receiving poor documentation, the test team or tester must understand the details and intricacies of the application, or have access to such knowledge to perform a thorough analysis in order to execute his/ her responsibility (Dustin, 2003).

APPLICATION / TECHNICAL KNOWLEDGE

Application software is the detailed instructions that control the operation of a system. The decision to develop or select appropriate software such as a complex domain system is a key decision, it is technology dependent and it has wide implications for the organisation.

For reliance, confidence and credibility computers and its software have to function consistently and above all, be error-free. The software is created and engineered by human beings and therefore not infallible. Quite the contrary, the development of software systems involves a series of activities in which the opportunities for interjection of human fallibilities are enormous (Deutsch, 1982).

Errors may begin to occur at the very inception of the process when the objectives of the software system may be erroneously or imperfectly specified, or later during the design and development stages when specifications are in the process of codification or being mechanized.

The application being technology dependent make bugs arising from it technology dependent as well – or at least influenced by the technology that the application was built with.

It is often helpful for testers and the team to understand how systems are built (Black, 2000).

Having members within the test team that have a grasp of the underlying programming languages, system architectures, operating features, networking skills, presentation and application navigation familiarity, database functional knowledge is a distinct advantage.

Severity and priority of bugs are driven by the business but requires adept and a deeper knowledge of the application and technical know-how to find them. Testers need to know how to find bugs effectively and efficiently.

By way of example, an ERP EB application with flexible choice driven by technology permits a combination of a number of benefits to employees (see Appendix C). Employees' desires are subject to changes based on their commitment, or on personal circumstances, social influences, economic conditions, legal status, and global or local political reform in order to choose a benefit structure that suits them at a point in time. The evolving (unstable) nature of this application makes for difficult test system processing, gives rise to ever changing test plans and test cases. Appendix C also provides a simplistic architectural view of an EB application. When domain knowledge is clearly insufficient, the test team requires specialist knowledge to be recruited on the team or gain access to people with such skills.

GENERAL PROFESSIONALISM

Testing requires disciplined creativity. Good testing, that is devising and executing successful tests – tests that discover the defects in a product – requires ingenuity, and may be viewed as destructive. Indeed, considerable creativity and competence is required to destroy something in a controlled and systematic manner (Kit, 1995). In addition, testers should be adept at reading, writing and general mathematics (Black, 2000). Test case development involves a thorough reading and comprehension of specifications, the business requirements, product and process documentation and a myriad of other references.

Whether trying to develop test cases or looking for bugs early in the SDLC, testers must understand what they read, grasp the goals and thereafter take the right actions.

It is critical that test resources must be able to write to gain the reader's understanding. This does not mean proper grammar or correct spelling, although poor language use may distract or confuse the reader. The most important issue is the ability to communicate effectively.

The application of metrics to ensure adequate coverage or to report the readiness or risk of the AUT may entail some degree or need for mathematical acumen.

What is required is an integrated approach to Software Testing that offers organisations the means to ensure that they are doing the right thing, at the right time, for the right reasons, and in the most effective manner possible.

Integration is a balance between creation and usage of appropriate knowledge. Balancing exploration (creating knowledge) and exploitation (sharing the knowledge internally) requires a well-developed internal transfer capability between functions. This requires a culture, reward systems, and communication networks that support the flow of knowledge and well functioning organisational memory to transcend time delays between developing and applying knowledge.

The general goal is to create a tangible manifestation of the team's knowledge. Ultimately, the team assumes responsibility for sharing its knowledge with the organisation at large.

An intelligent organisation bridges the knowledge of its domain experts, information content experts, and information technology experts (Choo, 1995). To be successful at software testing it is essential that the test team draws on the skills of information content experts, to work side by side with domain experts in collecting, analysing and designing test cases; and to act as consultants who transfer information after test execution.

Testing has become a profession – a career choice – and a place where knowledge workers make a mark. Software testing has evolved considerably, and has reached a point where it is

a discipline requiring trained professionals who get proper support from management (Kit, 1995).

From the literature it is clear that there is recognition of knowledge assets in software testing and the need to leverage these assets effectively. This is however tempered by the realisation that the path to achieving a knowledge approach involves complex adaptive systems, significant processes, mindset and culture change. It is unlikely that this change can be achieved in one giant leap, and a staged framework is desirable.

The Test Maturity Model (TMM) was developed by the Illinois Institute of Technology that contains a set of maturity levels through which an organisation can progress toward greater test process maturity. This model lists a set of recommended practices at each level of maturity above level 1. The aim is to promote a greater professionalism in software testing similar to the intention of the Capability Maturity Model (CMM) for software that was developed by the Software Engineering Institute (SEI).

MATURITY LEVELS in SOFTWARE TESTING

Building 'knowledge highways' and bridges to link isolated islands of domain, application / technical expertise and testing skills is a tough task and easier said than done. The TMM framework offers an informal an incremental knowledge approach – a behavioural view of an organisation that can be elicited at each level. It also provides hierarchical goals and a perspective on maturity growth and maturity deficiencies as shown in Figure 10:

- Level 1. – Initial
- Level 2 – Definition
- Level 3 – Integration
- Level 4 – Management & measurement
- Level 5 – Optimisation

Following Figure 10 is Table 2 which is populated using EB as a domain example based on the five levels - it shows grading maturity goals, sub goals, the set of activities, tasks and knowledge characteristics associated with each level.

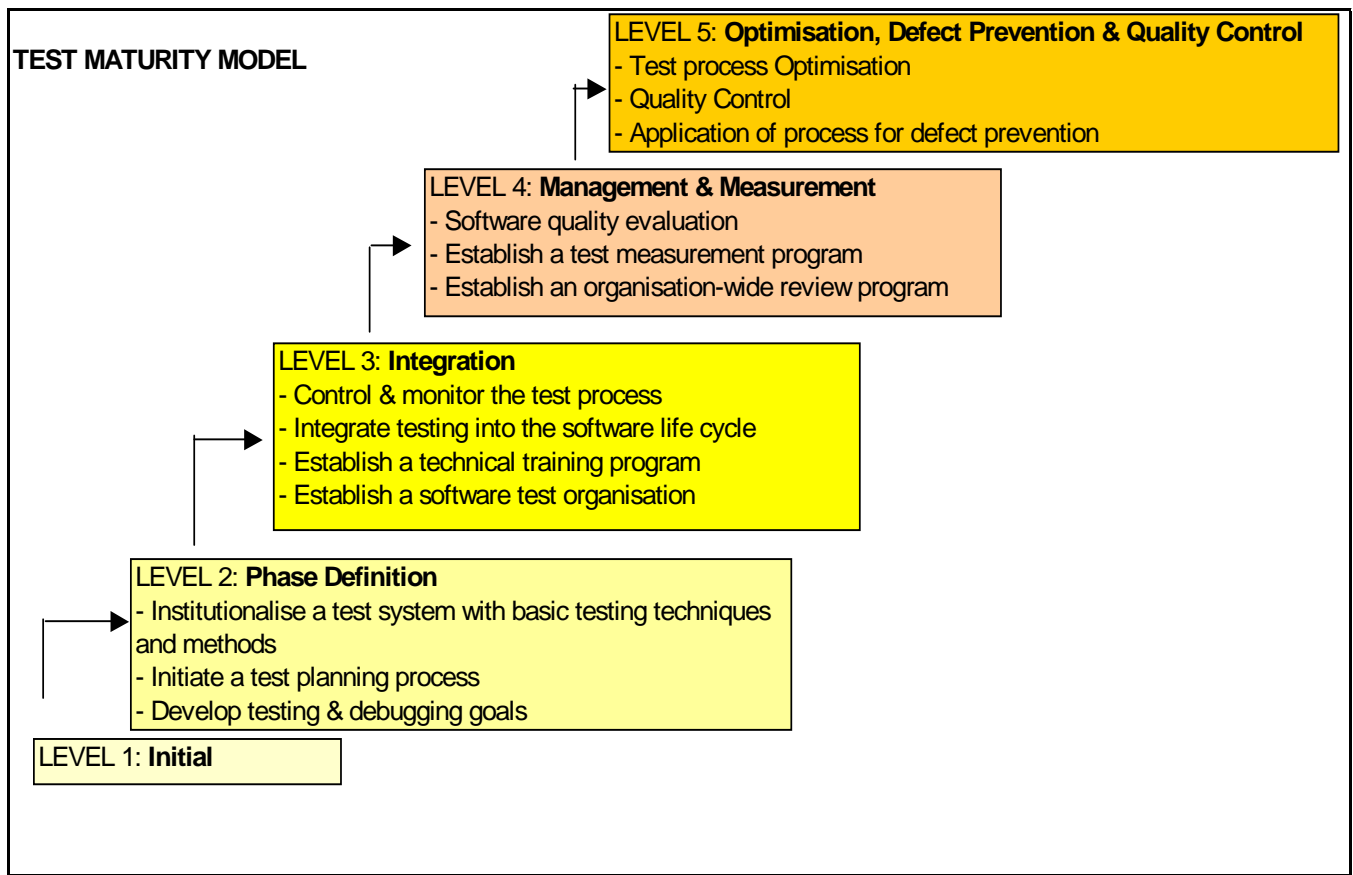


Figure 10: Test Maturity Model (Source: Burnstein et al, 2000)

The literature suggests that a mature software testing function include the following recommendations, standards, procedures and guidelines:

- A set of defined testing policies. A set of well defined and documented testing policies that are applied throughout the organisation.
- A test planning process. A well-defined and documented test planning process used throughout the organisation that allows for the specification of test objectives and goals, test resource allocation, test designs, test cases, test schedules, test costs and test criteria.
- A test life cycle. A well-defined test lifecycle with clear phases and test types throughout the cycle.

- An independent test team. The position of tester is defined and supported by upper management. Clear boundaries of responsibility and training opportunities exist and motivate test resources.
- A test process improvement group. A committed group of resources devoted to test process improvement. They can be part of a general process improvement group, or a software quality assurance group.
- A set of test-related metrics. The organisation has a measurement program. A set of test metrics is defined, data is collected and analysed for management reporting. The results are used in the test improvement process.
- A set of appropriate tools or the desire to acquire. Appropriate tools are available to assist the testing group with testing tasks, test management, to collect and to analyse test related data.
- Test process is tracked and controlled. The test process is monitored and controlled by the test managers to track progress, take corrective action when needed, to evaluate performance and capability, and to assess meeting the test criteria.

Level	(TMM) Software Testing Characteristics	Knowledge Goals to mature testing	Application-technical & Domain Knowledge characteristics in Test group (using EB as a Domain example)
1. Initial	<ul style="list-style-type: none"> • <i>Testing is chaotic and ill defined; not distinguished from debugging.</i> • <i>Test system & processes is adhoc or non-existent.</i> • <i>Software released without QA or reviews.</i> • <i>No tools neither a succession of competent resource pool.</i> • <i>Focus is largely on or only on positive testing.</i> 	<ul style="list-style-type: none"> • <i>No maturity goals at this level</i> 	<ul style="list-style-type: none"> • <i>Absence of the awareness to manage EB knowledge</i> • <i>All learning is reactive</i> • <i>EB knowledge in people's heads.</i> • <i>EB experts isolated pockets</i>
2. Definition	<ul style="list-style-type: none"> • <i>Software testing is separated from debugging and autonomously defined function in the SDLC.</i> • <i>Testing occurs too late in SDLC – post code test execution is still primary testing activity</i> • <i>Software Testing is a planned activity. Strongly based on the waterfall approach where testing follows construction phase.</i> • <i>Defects are found propagated from requirements and design phases.</i> 	<ul style="list-style-type: none"> • <i>Knowledge awareness</i> • <i>Basic training including knowledge of the fundamentals of testing</i> • <i>Software verification – it meets specifications</i> • <i>Develop testing and debugging goals for the delivery.</i> • <i>Assign responsibilities</i> • <i>Plan test phases. Test objectives, risks, test case analysis & design, resource allocation. Establish templates for each test type from unit,</i> 	<ul style="list-style-type: none"> • <i>EB knowledge reluctance to share or shared on as needed basis; mainly routine and discrete administrative tasks</i> • <i>EB basic education initiated</i>

		<p>systems through readiness testing.</p> <ul style="list-style-type: none"> Introduce test techniques and methods 	
3. Integration	<ul style="list-style-type: none"> Software testing is integrated into the SDLC and not a phase that follows construction. Test objectives and planning starts from requirements phase. Build on requirements from level 2 and continue supported by V-model type methodology Testing is regarded as important and a respected activity in the organisation. Specific Testing Methodology training identified Test measurement program not yet established 	<ul style="list-style-type: none"> Central knowledge database established Reviews are as yet informal Identified and established Test Training program. Funding and goals must be clear Test team or group established. Responsible for test system and the six key process areas³ Establish Test Training Campus Leadership support Clear roles and responsibilities Test team made up of competent contingent of resources Defined communication model between users, developers, testers and management Control and monitor Testing process 	<ul style="list-style-type: none"> EB knowledge life cycle visible and integrated into career EB knowledge is moderated and metrics collected Link between productivity and EB knowledge transference is visible Formal training established
4. Management and Measurement	<ul style="list-style-type: none"> Testing is measured and quantified process. Testing covers wider quality control activities such as reliability, usability and maintainability. Test case database & library established for reuse and regression testing. 	<ul style="list-style-type: none"> Abundant sharing of knowledge supported by management & organisational culture Reviews and inspection are structured and formal Test Training structured and strict compliance 	<ul style="list-style-type: none"> EB knowledge at a higher plane- at consultant level Impact of EB knowledge significant and demonstrable as productivity gains are achieved.

	<ul style="list-style-type: none"> Defects are logged, managed and tracked with proper severity classification. Reviews across all phases now recognised as testing and quality control activities. Non-functional testing is an integral part of software test phases. 	<ul style="list-style-type: none"> Establish an organisation wide review program Establish test measurement program for full test life cycle QA group Integrated into and part of the organisational culture 	<ul style="list-style-type: none"> Feedback loops are qualitatively better and tighter as domain knowledge increases Mature 'EB knowledge market' exists
5. Optimisation	<ul style="list-style-type: none"> Testing is strategised in the SDLC and continuously improved upon. Established procedure, everybody understands it and full compliance achieved. Defect prevention and quality control practices throughout the SDLC. Automated tools, fully supported, facilitate Test system and throughout the test cycle. 	<ul style="list-style-type: none"> Knowledge optimized and reused Defect prevention team established Visible defect prevention Management reporting – established frequency Quality control in place across SDLC with focus on test activities. Implement improvement practices Tracking and report improvement processes Process improvement group established 	<ul style="list-style-type: none"> Boundless EB knowledge in team. Boundaries are irrelevant EB knowledge ROI integral to decision making Continuous improvement and sustaining domain knowledge EB knowledge is of such a level, the ability to shorten test cycles by shaping and influencing change.

Table 2: Test Maturity Model adopted for a typical domain area (Source: Burnstein et al, 2000)

SUMMARY

- Critical knowledge can be regarded as the sum of the know-how and the skills that are shared within a test team particularly in the areas of software testing, domain knowledge, application-technical expertise and general professionalism.
- Testing requires disciplined creativity. Good testing, that is devising and executing successful tests – tests that discover the defects in a product – requires ingenuity, and maybe viewed as destructive. Considerable creativity and competence is required to ‘destroy’ something in a controlled and systematic manner. Testing has become a profession – a career choice – and a place where knowledge workers make a mark.
- Test skills requirements will differ from project to project.
- Domain knowledge in the test team could be an advantage if projects are pressed for time and perhaps a disadvantage if projects are quality sensitive. The trade-off between quality, coverage and costs are determined by the amount of competent resources, schedule and budget that the project can tolerate. While recognising the importance of domain knowledge, the literature is not conclusive about the extent of domain knowledge within a test team and how it will influence the quality of the AUT.
- Application-technical skills in the team are distinctly advantageous.
- General professionalism, attitude, the ability to read adeptly, write clearly and have a mathematical bias are certainly favourable additions to the test team.
- The recognition of knowledge within software testing is tempered by acknowledging that in order to leverage knowledge assets effectively, the organisation will have to change in terms of process, paradigm and culture. The TMM introduces a practical framework that offers organisations a progressive grading of software testing through various levels of maturity.

CHAPTER 6

DATA ANALYSIS

INTRODUCTION

This chapter presents an overview of the empirical data collected and relates these observations to the literature (the source of the questions constituting the survey instrument) as reviewed and summarized in the previous chapters.

As indicated in the foregoing chapters the literature presents various knowledge models based on four KM enablers, *leadership*, *culture*, *technology* and *measurements* that act in a dynamic relationship with KM processes. The emphasis for this research is placed on interrelationships between people in software testing in particular knowledge sharing, demographic factors that may predispose individuals to knowledge sharing, the requisite knowledge and the organisational culture that promotes quality delivery through software testing.

The technique of collecting data through interviews and a survey instrument (see Appendix E) started with a select target group of resources known to the author across various organisations including professional software test trainers, tool and test service providers. A second group of data resulted from the primary target group that forwarded the questionnaire, to colleagues, employees and seniors.

The author invited eight participants to participate in semi-structured interviews. These interviews were largely based on the questionnaire previously sent and completed by the interviewees. The purpose was to gather additional data, clarify areas of uncertainty within the questionnaire, confirm and reinforce the data collection and refine the survey.

It should be noted that while this approach may appear to be biased, the participants' responses were in no way influenced by the author and all data is reported as it was recorded.

The planned total number of participants was 55, based on 5 referrals per organisation across 11 organisations. The total number of referrals received was 72 from 8 organisations, the final number of responses received by email was 40.

Six questionnaires were discarded due to poor quality and non-compliance with the selection criteria. Emails were sent to participants, in an attempt to complete the non-compliance and missing data, as telephone contact details were not provided. This proved unsuccessful. The four vendor respondents were excluded to distinguish testers from suppliers thereby providing a more consistent sample. The final total sample population ended up as 30 participants.

Given the relatively small number of participants no valuable or statistical significance should be read into the tables that follow. Further research into this topic is warranted.

DATA GATHERED

The survey instrument synthesized a priori judgmental knowledge to focus on three areas:

- Demographics (personal and situational factors)
- Respondents attitude to knowledge sharing, and
- Assessing/rating the (status) opinions of the maturity levels of software testing within the organisation

DEMOGRAPHIC DATA

TYPE OF EMPLOYMENT

This variable was split between permanent employment and contract employment. The distinction was made to gauge if the type of employment has any bearing on knowledge sharing in software testing. Only with respect to consulting companies was the data further defined in terms of consulting in different sectors as opposed to a single environment.

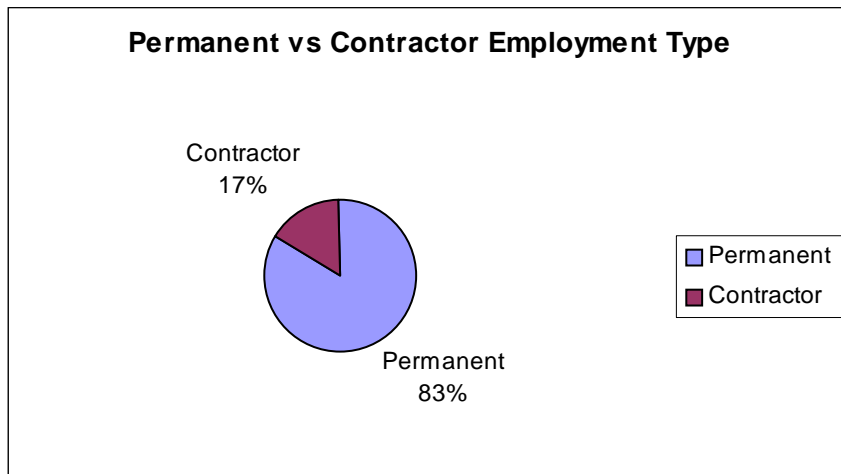


Figure 11: Sample population – Type of Employment

The above diagram shows that 5 respondents are contractors while 25 were permanently employed.

COMPANY SIZE

Company size was tested according to the company that employed the respondent, not the company that the respondent was consulting at.

Size	Description	Count	% of Population
Small	< 30 employees	6	20
Medium	31 – 100 employees	6	20
Large	101 – 250 employees	3	10
Corporate	> 250 employees	15	50
Total		30	100

Table 3: Company Size

The table above shows that a greater part of the sample population (50%) worked in a corporate environment. This is consistent with the specialized nature of software testing.

AGE AND GENDER

These demographics were requested to assess if there was any influence on the overall attitude to knowledge sharing. The population profile emerged as follows:

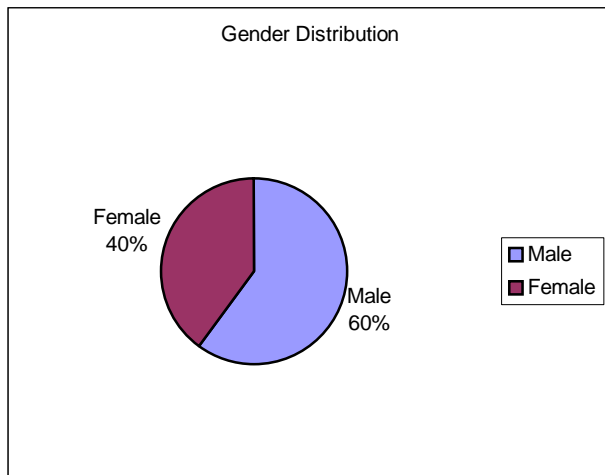


Figure 12: Gender distribution

Of the sample population 40 % (12) were female while 60% (18) were male. It is not possible to glean from the literature whether this in anyway reflects the greater software testing population.

The details of the age demographic showed a mean age of 35.66.

Youngest	Mean	Oldest
28	35.66	57

Table 4: Age statistics

The age details are consistent with the economically active population and organisations surveyed.

EDUCATION AND TRAINING

This section of the data gathering process covered tertiary qualifications and software testing training. It was found that 80% (24) of the sample population had tertiary education while only 27% (8) had formal training in software testing.

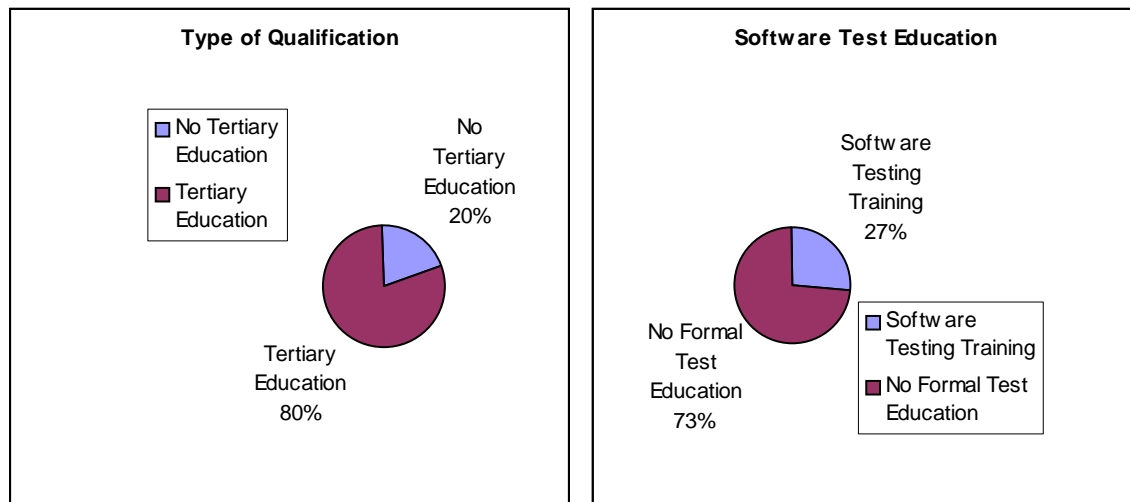


Figure 13: Categories of tertiary and Software Training.

WORK EXPERIENCE

The third area of demographic data covered work experience spanning Total years, IS and Software Testing experience.

Type of experience	Lowest number of years	Mean	Highest number of years
Total years experience	6	20.2	48
IS	1	13.88	38
Software Testing	0	7.98	30

Table 5: Work Experience

The results from the sample population as shown in table 5 above indicates a relatively youthful software testing industry despite the total work years of experience. A significant gap between Software testing and IS years corresponds with the literature. Beizer, Paul et al, Dustin state that historically across the SDLC resources covered every aspect from business requirements, analysis, construction, testing and implementation (and perhaps even

support). This has a direct relationship with the age variable discussed previously. As stated in chapter 4 a new era of software testing, a vocation of specialization is emerging.

ATTITUDE TO KNOWLEDGE SHARING

The second focus area in the survey instrument was aimed at measuring the respondent's attitude to knowledge sharing. This was achieved by a 5-point Likert-scale ranging from 'strongly agree' to 'strongly disagree'. Table 6 gives an indication of the questions as well as the responses received (based on the mean response value per question). It is clear from the profile that the majority of the responses fell within the uncertain range of 'neither agree nor disagree', indicating infancy or developing phase of knowledge sharing in the population. However, it is interesting to note that there is a strong encouragement and culture within organisations to share knowledge.

Question: • (Refer to Appendix E- Survey Instrument for the full list of questions asked)		Strongly Agree	Agree	Neither Agree nor Disagree	Strongly Disagree
13	I follow a formal methodology in Software Testing.			•	
14	I regularly use Software Testing tools / software.				•
15	I regularly use templates.				•
16	Are you familiar with the concept of a Test Life Cycle?				•
17	I regularly reuse existing knowledge when performing my test function.		•		
18	There are often situations where I am able to share my test specific knowledge (formal or informal)			•	
19	Knowledge sharing is a high priority with respect to testing in my organisation.			•	
20	I regularly maintain a personal knowledge base.			•	
21	I contribute to a corporate knowledge base.				•
22	When I am faced with a new project in an area I have not worked in before, I will first use a knowledge base or draw on the knowledge of others in this area before starting.			•	
23	In an attempt to reduce my test risk, I use a knowledge base or draw on the knowledge of others in this area.			•	
24	The culture of my current organisation does not encourage knowledge sharing.				•

Table 6: Attitudinal response profile

The sample population was also asked to rate their attitude to knowledge sharing as follows:

*How would you rate your personal attitude to knowledge sharing? (Tick the **most appropriate** box)*

- ☐ *I will actively look for opportunities to share my knowledge and that of others*
- ☐ *If opportunities arise that require me to share my knowledge I will do so willingly*
- ☐ *Knowledge sharing is a good idea but I don't find time to do it*
- ☐ *I tend to oppose knowledge sharing*
- ☐ *I am strongly opposed to knowledge sharing*

Of the respondents 90% (27) answered this question positively, selecting either the first or second option whilst 10% (3) agreed that knowledge is a good idea but did not find the time to do it. This presents us with 2 distinct groups of respondents - those that are classified as 'active knowledge sharers' ticked the first option (57%), and 'passive knowledge sharers' (33%).

BENEFITS AND DRAWBACKS OF KNOWLEDGE SHARING.

As gleaned from the literature, for the important risk areas in software testing, this section used rating (critical to not relevant) as a gauge to assess benefits and drawbacks. The data gathered was summarized as follows:

BENEFITS

Factor	Rating				
	Critical	Very Important	Important	Somewhat Important	Not Relevant
1. Saves Time	5	18	5	2	0
2. Improves quality of the solution	15	9	6	0	0
3. Synergistic value	4	12	8	6	0
4. Decrease costs	14	7	5	4	0

Table 7: Benefits of Knowledge sharing

Table 7 shows that the perceived benefits were seen to be either critical or very important by the majority of respondents.

PERCEIVED DRAWBACKS

Factor	Rating				
	Critical	Very Important	Important	Somewhat Important	Not Relevant
1. Too time consuming	8	5	7	5	5
2. High cost of implementation	1	8	7	7	7
3. Tools are inadequate to support knowledge sharing	1	7	9	6	7
4. Do not get credit for sharing knowledge	1	7	8 8		6
5. Loss of intellectual ownership	7	4	5	4	10

Table 8: Perceived Drawbacks of Knowledge sharing

Table 8 shows that, while a good percentage of the sample population rated certain perceived drawbacks as 'not relevant', or 'somewhat important', a good proportion of the respondents felt that time, cost, tools, getting credit as well the risk of losing intellectual ownership ranged from being 'critical' to 'important'. It is interesting to note that the responses to benefits of knowledge sharing were decisive and strongly supported, while by the same sample population there was uncertainty about the perceived drawbacks. This may indicate that whilst the sample population perceived the benefits of knowledge sharing to be an advantage, management and controls of the identified drawbacks, if left unattended, may prevent spontaneous (informal or formal) knowledge sharing.

FACTORS INFLUENCING SOFTWARE QUALITY

The respondents were also asked to give an opinion on factors that influence quality of software.

This section used a combination of ranking questions (most important to least important) and rating (critical to not relevant). The data gathered was summarized as follows:

Factor	Importance			
	Most	Fairly	Medium	Least
Application Domain knowledge	17	6	5	2
Software Testing competency and knowledge	5	10	9	6
Credibility of source	8	12	4	6
Timely engagement of testing in the SDLC	2	11	8	9
Maturity level of testing and flexibility to adapt existing knowledge	3	13	5	9

Table 9: Factors affecting quality of delivery

From the mean sample population the ranking of these factors are in the following order of priority:

1. Application domain knowledge
2. Maturity level of testing and flexibility to adapt existing knowledge
3. Credibility of source
4. Timely involvement/engagement of testing in the SDLC
5. Software Testing competency and knowledge

This is a corresponding concern expressed in the literature where it was felt that software testing is erroneously regarded as negligible or peripheral to domain and technical knowledge.

OWN KNOWLEDGE VERSUS ACCESS TO KNOWLEDGE

The next section tests Thomas' assertion that there are two types of knowledge that is important; the first is our own, and the second we should know where to find and have access to it.

Factor	Rating					
	Critical	Very Important	Important	Somewhat Important	Not Relevant	
1. Lack of domain knowledge	15	10	3	2	0	
2. Lack of testing knowledge	4	9	9	6	2	
3. Lack of application knowledge	8	8	11	1	2	
4. Lack of access to appropriate domain knowledge	8	10	6	4	2	
5. Lack of access to proper testing knowledge	4	6	10	5	5	
6. Lack of access to application knowledge	4	12	5	3	5	

Table 10: Own versus Access to knowledge

From the mean sample population the ranking of these factors are in the following order of priority:

1. Lack of domain knowledge
2. Lack of access to application or technical knowledge
3. Lack of access to appropriate domain knowledge
4. Lack of testing knowledge
5. Lack of application knowledge
6. Lack of access to proper testing knowledge

There are areas of overlaps and areas of dual importance presented by the sample population. The above corresponds with the literature when it states that identified domain, application and testing skills are all important in concert and their combination competencies will vary from project to project depending on various factors such complexity of the delivery under test (see chapter 4 -Black, Dustin, Kit).

DOCUMENTS PRODUCED BY TEST ENGINEERS

The respondents were asked on the types of documents they usually produced. Table 11 shows a summary of the respondent's answers

Document	Count	% Of Cases
Test Strategy	15	50
Test Approach	9	30.0
Test Plans	20	66.7
Test Decision Tree	8	26.7
Requirements Matrix	18	60.0
FMEA (Failure Mode and Effect Analysis) spreadsheet	1	3.3
Risk Assessment profile	5	16.7
Test Priority ranking	7	23.3
Test schedule	18	60.0
Test Summary reports	7	23.3
Application Readiness reports	3	10.0
Checklists	16	53.3
Other	3	10.0

Table 11: Documents completed by Test engineers

The mainstream literature shows that a large component of knowledge stock and capability is in making tacit knowledge explicit formally through documentation or templates. An influencing and impacting factor is the creation and usage of these documents. From the results of Table 11 it is evident that a number of knowledge stock areas are in the early stages of development and must be read in association with the informal approach to testing (see Choo Chapter 3, and Chapter 6, also Gallagher Chapter 6).

AREAS TO COLLECT AND SHARE DATA WITHIN SOFTWARE TESTING DISCIPLINE

Respondents were asked to identify areas they felt were important to collect and share data within software testing. They were also asked what value would this add and whether it was an organisational practice currently. The salient data collected is summarised as recorded in the survey instrument in Table 12

Area	Value Contribution
1. Knowledge of the system	It contributes to the quality and scope of testing
2. Construction of test cases that cover full scope	It contributes to the quality and scope of testing
3. Maturity of testing	Access to knowledge and history to evolve to a higher level
4. Business/ domain knowledge	It aids in familiarisation for new recruits.
5. All areas – any tester would need to improve their knowledge in any area which is weak – be it technical, business, testing techniques	Improves testing output

Table 12: Areas of value to collect and share within Software Testing

Many respondents noted that these areas of knowledge sharing in software testing within their organisation was either happening informally or inconsistently

TEST MATURITY LEVELS

The last focus area in the survey instrument was aimed at gauging the respondent's opinions of the software testing maturity level in their organisations - a current state. This was achieved by a 4-point Likert-scale ranging from 'Very Adequate' to 'None'. Table 13 gives an indication of the questions as well as the responses received (based on the mean response value per question).

Criteria <ul style="list-style-type: none"> (Refer to Appendix E- Survey Instrument for the full list of questions asked) 	Very adequate	Adequate	Inadequate	None
Adequacy of formal training in software testing			•	
Adequacy of equal sharing of success and failure across the team		•		
Adequacy of the tester's inclusion in the entire SDLC		•		
Adequacy of the tool sets available to testers before execution.			•	
Adequacy of training and support testers receive re tools			•	
Adequacy of management's time and attention to testing			•	
Adequacy of management's commitment and investment			•	
Adequacy of the profiles that testers have of Domain knowledge			•	
Adequacy of the testers' interaction with all levels of users		•		
Adequacy of the test management involvement			•	
Adequacy of communication			•	
Adequacy of response times, acceptance and resolution			•	
Adequacy of the definition of the requirements and of the design for developing test conditions			•	
Adequacy of the documentation of changes			•	
Adequacy of the shared understanding across the stakeholders			•	

Table 13: Test Maturity level in organisations

Table 13 shows a very low level of software testing maturity as perceived by respondents of their own organisations.

The final question asked of respondents, given their 'maturity gauge', was to provide an opinion as to whether the organisation on average implemented more failures or successes. (Refer Appendix E - Survey instrument for the definition of failure as per the questionnaire).

The results are summarised in Table 14

	Count	% Of population
Failures	7	23%
Successes	23	77%

Table 14: Perceived Failure-Success rate

It is interesting to note that an overwhelming success rate of 77% (23) is recorded despite the low level of maturity perceived. This appears to be inconsistent with the maturity models provided by the literature.

CHAPTER 7

FINDINGS AND CONCLUDING THOUGHTS

INTRODUCTION

The previous chapters built the topography, the content and context of the components of knowledge, software testing, domain area and application/technical expertise explaining the extent of the literature reviewed and fieldwork conducted. More specifically chapter 3 and 4 focused extensively on the secondary sources (literature) to respond to hypotheses 1 and 2, whilst chapters 5 and 6 using descriptive empirical analysis techniques (Mouton, 2001) focused on fieldwork and a survey instrument (primary sources) to respond to hypotheses 3 and 4.

Whilst the majority of the sample respondents were randomly selected, the author is acquainted with a specific smaller target group. It is therefore not reflective and not possible to extrapolate the results of this research to the greater test community, however, the results is a good basis for further research and to give an indication of what may be expected if a random sample is used.

Please also note that the sample size may have a bearing on the findings. As Saunders states that it is very difficult to obtain a significant statistic with a small sample (Saunders et al, 2000). The sample size of 30 used in this research qualifies the findings and must be considered when reviewing or assessing the results.

This research set out:

- To assess if demographics (personal or situational) predisposes test engineers to sharing knowledge.
- To investigate the extent of knowledge sharing in the discipline of software testing.

- To establish if domain knowledge per se influences the quality of effective software testing.
- To establish if application/technical knowledge of the system under test influences the quality of effective software testing.

The findings of the literature study follow the empirical research in the sections below.

PERSONAL FACTORS INFLUENCING KNOWLEDGE SHARING

While 7 factors were identified (age, gender, experience, tertiary education, software test training, company size and role type (permanent or contractor)) as potential areas of association with a specific attitudinal response, it is not possible to conclude that there is significant association between any of the personal factors tested and test engineers' predisposition to knowledge sharing. Therefore, it cannot be stated conclusively from the evidence that any significant association between personal or situational factors and the test engineer's attitude to knowledge sharing.

EXTENT OF KNOWLEDGE SHARING

The mainstream knowledge thinkers such as Nonaka, Choo, Davenport, Prusak, Sveiby and other authors identified various factors of significant importance to knowledge creation, knowledge sharing or knowledge enabling – some of these were tested with the survey instrument.

ENABLING - ATTITUDE AND WILLINGNESS

A direct question asked of the respondents was to rate their attitude to knowledge sharing.

The responses showed that all participants would share their knowledge willingly. The only differentiator was that 57% of respondents said they would actively look for opportunities to share knowledge, while the remainder would share knowledge when called upon to do so.

Question 26 asked respondents if they thought knowledge in software testing was valuable.

This question received a 100% positive response. It is evident that the value, desire and

willingness to share knowledge is recognized and given the correct circumstances and motivation, every test engineer in the sample population will do so.

KNOWLEDGE STOCK

Questions that tested the extent of knowledge sharing included a question on 'valuable areas to collect and share data' within software testing. The respondents were also asked whether this was currently done in their organisation. Results of the sample population (more detail in chapter 6) showed a positive response to sharing and to provide meaningfully to areas where knowledge in software testing can make a difference.

In addition to this, question 21 asked respondent to rate their contribution to a corporate knowledge base. Results showed that 16% strongly agreed. This may indicate that while test engineers have a positive attitude and see value in knowledge sharing, the current preferred practice appears to be informally rather than through a formal knowledge base approach.

From the responses received from the question on knowledge database access or usage of templates and documentation, it is evident that a number of knowledge stock areas are in infancy. This must be read in association with the informal approach to software testing.

RISK REDUCTION

Choo (see chapter 3) mentions three distinct areas in which the creation and use of knowledge play a strategic role in determining an organisation's capacity to grow and adapt. One being usage of accumulated knowledge.

Participants were asked to rate their use of knowledge base or their likelihood of drawing on the knowledge of others in an attempt to reduce project risk. The results showed that 76% of the sample population would draw on a knowledge source in order to reduce risk.

However, existing practice from the sample population also shows that 50% are indifferent – they are not concerned about retaining a personal database. Added to this, the response of more than 70% did not contribute to a corporate knowledge base. This reinforces the belief that the current knowledge practices as being riskier, informal or ad-hoc in its approach to software testing.

ORGANISATIONAL CULTURE

The organisational culture appears to provide for ample opportunity to share knowledge. This is supported by the response to question 18 that asked respondents about situations to share knowledge – this question received a 70% positive response. And added to this, is question 24 that asked if the organisation encourages a knowledge culture – this question received a 63% positive response.

BARRIERS TO KNOWLEDGE SHARING

The literature tells us that knowledge is not diminished if you share it and the most common barrier to knowledge sharing is the ‘misunderstanding that sharing knowledge will lead to reduction of personal power’ (Sveiby, 1977). This concept was tested in the survey instrument where participants were asked to rate the perceived drawbacks of knowledge sharing. With regard to loss of personal power or intellectual ownership, 36% of the respondents thought this to be critical or very important whilst 33% felt this factor is irrelevant as a barrier to knowledge sharing. Therefore from the sample population it is not conclusive and no clear definitive qualification can be placed on the reduction of personal power due to knowledge sharing.

The sample population identified the option ‘too time consuming’ as the most important drawback.

BENEFITS OF KNOWLEDGE SHARING

The literature identified various benefits such as time saving, cost saving, test team synergy and better quality solution as a direct result of knowledge sharing. The sample population rated all the benefits as ‘critical’, ‘very important’, ‘important’ or ‘somewhat important’ – none of the respondents felt any of the factors to be irrelevant. The most important benefit of knowledge sharing emerged as ‘improved quality’ (50% ranked this as critically important) while ‘time savings’ (60% ranked this as very important) was considered the next most important benefit.

All of the above supports the position that knowledge sharing, the willingness to share and the extent of sharing it in organisations as quite significant in the sample population. Whilst it is clear that the sample population finds value in knowledge sharing, as stated earlier, it is not possible to state conclusively that this may be reflective of the greater population due to the quantity of respondents and method of sampling employed.

FACTORS INFLUENCING QUALITY OF DELIVERY IN SOFTWARE TESTING

Stewart Thomas quotes Samuel Johnson as saying that knowledge is of two kinds; we know a subject ourselves, or we know where we can find information upon it. Software testing is actually practicing epistemology (Theory of the method or grounds of knowledge - Bach et al, 2001).

The literature is clear that the quality of software testing is affected by certain kinds of knowledge such as:

DOMAIN KNOWLEDGE

Authors indicate that there are many people who never bothered with domain knowledge that have accomplished good results, but the literature also tells us that if you want *better than good*, commit to learning it. Studying the domain area will help you devise effective testing strategies, better recognize mistakes in your work, know what your testing does and does not prove, and construct defensible test reports.

This corresponds with the survey where the majority of respondents (83%) placed the 'Lack of domain knowledge' in the category of critical or very important, while 60% also rated the 'Lack of access to domain knowledge' as critical or very important in influencing the delivery of quality in software testing.

SOFTWARE TESTING COMPETENCY

Testing, in general, is conducted to verify that software meets specific criteria and satisfies the requirements of the end user. Effective testing uses its competency area to ensure that the AUT will function correctly under all circumstances. Essential to achieving this is the

detection and removal of defects in the software. In addition to knowing where bugs live and what they look like, it is critical that testers are equipped and knowledgeable about how to find them, effectively and efficiently. The testing process is an important subset of the overall software development process; therefore, its methods and maturity growth needs support from key process areas associated with general process growth in the SDLC (see KPAs chapter 4).

In assessing 'Lack of testing knowledge' as a factor in delivering quality, the sample population showed that 33% thought it either critical or very important, 30% felt that is important, 20% somewhat important whilst 6% thought it irrelevant.

The literature describes testing as an emerging specialist area, to be regarded as the future currency of software quality.

APPLICATION OR TECHNICAL EXPERTISE

Since software testing is about tools, about test architectures, test systems, about data, analysis and interpreting and reporting on bugs. These are often technology dependent, or influenced by it. Therefore it is regarded a distinct advantage for testers to understand how systems are built. As supplementary value, is having a grasp of the application, system architectures, operating system features, presentation layers, database functionality or the technical environment in which testing will be conducted or the tested application will be deployed.

EDUCATION AND TRAINING

Test engineers with tertiary qualifications are advantaged. However, we do have problems specific to KM and software testing, for example:

- There is no generally accepted standard body of knowledge or center of competency moderation for testing
- Software testing is not widely taught in colleges or universities
- KM is relatively new and also not widely found in colleges or universities.

- A knowledge approach to any phase in the SDLC including testing is not readily found in the literature.

KNOWLEDGE OF THE SDLC AND VARIOUS MODELS

Knowledge of what constitutes successful end-to-end testing effort is typically gained through experience. The realization that a testing program could have been much more effective had certain tasks been performed earlier in the life cycle is valuable knowledge. Early or timely engagement of software testing in the SDLC (see Figure 5 V-model chapter 4) not only supports effective test design, which is a critically important activity, it also provides for early detection of errors and prevents migration of errors from requirements specification to design, and from design to code. This kind of error prevention reduces cost, minimizes rework, and saves time. In addition to knowledge capability, reviews and inspections, the earlier in the cycle that errors are uncovered, the easier and less costly they are to fix (Refer to Appendix F – tabulates the increasing cost of error remediation as progress is made to the latter phases of the SDLC).

IN SUMMARY, A KNOWLEDGE APPROACH TO SOFTWARE TESTING:

- Given the sample population and method employed the results did not show a significant association between any of the demographic factors (personal or situational).
- The extent of knowledge sharing amongst the testing fraternity shows evidence of an established informal knowledge sharing culture where the sample population either presented no restraint in sharing, quite the contrary, they expressed their willingness to share their knowledge.
- Both the literature and the sample population concur that software testing may be accomplished without extensive formal software testing skills. However, structured testing competency and knowledge of the application or the domain area transcends mediocrity -

it is a distinct advantage to expand knowledge inclusive of the domain and application/technical areas.

Whilst a thorough knowledge of software testing or being test proficient per se is recognised as a factor that impacts the degree of software quality proportionately, the literature also advises gaining disciplines of general tertiary education as an important and distinguishing differentiator.

CONCLUDING THOUGHTS

This research set out to establish if domain and application/technical knowledge or the interrelationship of team dynamics has any bearing on the quality of effective software testing, and to determine if any personal or situational factors would predispose the test engineer to knowledge sharing.

The purpose of this study was to conduct research into the best practices and current trends regarding knowledge sharing as adopted by practicing and professional test engineers. It covered existing literature pertaining to software testing and KM and included, by way of an example, documentation of a specific domain area, namely EB.

A survey instrument was designed to gather data from a sample population.

The inferential analysis done on the sample data was not conclusive and did not show a significant association between any personal or situational factors that may predispose the test engineer to knowledge sharing.

The results showed a significant willingness to share and apply knowledge. However, from the sample population no evidence of a formal or structured knowledge sharing approach in software testing was found.

It is not desirable to have a testing team made up exclusively of knowledge expert testers with years of testing experience. The most effective software testing will be achieved by a team that consist of members with a mixture of expertise such as subject matter (domain knowledge), technology and application knowledge, good testing techniques as well as

general tertiary type discipline - this mixture, should be dispersed with a varied degree of experience levels from beginners to expert resources.

A knowledge approach is the themes and variations in them that organize the experience of team structures that can be found in the interactions and interrelationships among people. It is the commitment of the team to excellence, their mastery of their crafts and their ability to work together that is the essence of a quality delivery, not the rules or policies that govern them.

TEXT REFERENCE NOTES

1. 'Best' practices imply a completion of improvement, the ultimate. This can be no further from the truth when we evidence continuous improvement. Snowden prefers the term 'past practice'. Alternatively it is a practice of a point in time and certainly not to be confused as 'best'. Usually best practices in the literature are associated with those practices that have produced outstanding results in another situation that can be adapted.
2. It is standard research practice to distinguish between primary data and secondary information sources.
 - Primary sources refer to *data (not previously analysed or interpreted)*: whether you have to collect it yourself (e.g. by way of the survey instrument) or whether it already exists in one form or another.
 - Secondary *information* sources refer to written sources of information (including the internet) that discuss, comment or debate and interpret information – this source is a manner of gaining from another (informed) opinion about a topic (e.g. literature review).
3. Silo - in organisational terms refers to an autonomous unit operating independently - in its own right and usually associated with unwillingness to share across the organisation.
4. Pervasive is defined as spread throughout. By pervasive it is meant that testing does not start and stop by strict adherence to clinical time-frames as the traditional waterfall method would suggest in the SDLC.

The waterfall approach breaks the development cycle down into discrete phases each with a rigid sequential beginning and ending. The waterfall model suggests that each phase should be fully completed before the next is started. And once a phase is completed, in theory one never goes back to it.

REFERENCES

- Baba I. 2004. Personal Communication. 19 April, Pinelands, Cape Town.
- Bach, J. Kaner, C. & Pettichord, B. 2002. *Lessons learned in Software Testing. A context driven approach*. New York: John Wiley & Sons
- Ballou, Melinda-Carol 2002. *Coordinating Testing Phases and personnel. Application delivery strategies*. Meta Group: Stamford Press.
- Barker, J.A. 1993. Paradigms. *The business of discovering the future*. New York: HarperCollins Publishers.
- Beizer, Boris 1984. *Software System Testing and Quality Assurance*. Macmillan of Canada: Van Nostrand Reinhold Company Ltd
- Beizer, B. 1995. *Black-box Testing: Techniques for Functional testing of software and systems*. Canada: John Wiley & Sons.
- Beizer, B. 2001. *Software is different*. Quality Techniques Newsletter. Part 3 of 3. San Frisco. US. [online].
- Available: <http://www.qtn@soft.com>
- Bender, R. date unknown. *How do you know when you are done testing? Gartner Research article*
- Black, R. & Baumann, T. 1999. *Test Execution Process*. Microsoft Press. Boston: US.
- Black, R. 2002. *Investing in software testing. The importance of the right technique*. A Gartner Research published article.
- Black, R. 2002. *Managing the Testing Process*. New York: Wiley Publishing
- Black, R. 2004. *Critical Testing Processes. Plan, Prepare, Perform, Perfect*. New York: Wiley Publishing.
- Burnstein, I., Suwannasart, T. & Carlson, C.R. 1996 – 1999. *Developing a Testing Maturity Model : Part I & II*. Illinois Institute of Technology. Chicago. US.

- Castells, M. 1996. *The Rise of the Network Society. The Information age: Economy, Society and Culture*. Malden Massachusetts: Blackwell publishers. Vol I.
- Chafeker, I. 2004. Personal Communication. 28 April, Pinelands, Cape Town.
- Chillarege, R. 1999. *Software Testing Best Practices*. IBM Research. Technical report. Reference number:RC 21457 log 96856
- Choo, W.C. 1995. *Information Management for the Intelligent Organisation: Roles and implications for the Information Professions*. Paper presented at the Digital Libraries Conference 28 March 1995
- Choo, W.C. 1998. *The Knowing Organisation*. New York: Oxford University Press.
- Choo, W. C. 200. *Closing the cognitive gaps: how people process information*. In Financial Times: Mastering Information Management. Complete MBA companion in information management / Edited by T. H. Davenport, T. Dickson & D. A. Marchand. Harlow UK: Prentice Hall.
- Craig,R. 1999. Test Strategies and Plans. Software Quality Engineering Inc. Ornage Park. Florida US.
- Crosby, P.B. 1984. *Quality without tears. The art of hassle-free management*. New York: McGraw-Hill
- Cronje, T. 2004. Personal Communication. 26 April, Compuware V&A Waterfront, Cape Town.
- Davenport, T. H. 2000. *Mission Critical. Realising the promise of Enterprise Systems*. Boston. Harvard Business School Press
- Davenport, T. H. & Marchand, D. A. 2000. *Is KM just good information management?* In Financial Times: Mastering Information Management. Complete MBA companion in information management / Edited by T. H. Davenport, T. Dickson & D. A. Marchand. Harlow UK: Prentice Hall.
- DeMarco, T. & Lister, T. 1999. *Peopleware. Productive Projects and Teams*. New York: Dorset House Publishing company.

- Despres, C & Chauvel, D. 2000. *How to map knowledge management*. In Financial Times: Mastering Information Management. Complete MBA companion in information management / Edited by T. H. Davenport, T. Dickson & D. A. Marchand. Harlow UK: Prentice Hall
- Deutsch, M.S. 1982. *Software verification and validation: Realistic Project Approaches*. US: Prentice-Hall.
- Drucker, P. F. 1993. *Post Capitalist Society*. New York: HarperCollins
- Dustin, E. 2003. *Effective Software Testing. 50 Specific ways to improve your testing*. Boston: Addison–Wesley.
- Ebrahim, I. 2004. Correspondence. February - 30 April, Liberty Gauteng.
- Ehms, K. & Langen, M. 2002. *Holistic development of Knowledge Management with KMMM*. Siemens AG
- Fitz-enz, J. 1993. *Benchmarking Staff Performance*. New York: Maxwell Macmillan International Publishing
- Fewster, M. 2001. *Common mistakes in Test Automation*. Grove Consultants.
- Gallagher, S. & Hazlett, S. 2000. *Using The Knowledge Management Maturity Model (KM³) As an Evaluation Tool*. University of Belfast . BT7 1NN.
- Ford, B. 2004. Personal Communication. 26 April, Kenilworth, Cape Town.
- Gard Analytics. 2003. *Building simulation software testing*. [online].
Available: <http://www.gard.com/softwareTesting.htm>
- Gartner Research. 2002. *The Software Test architect currently unknown*
- Gates, W.H. 1995. *The Road Ahead*. New York: Penguin Books
- Gates, W.H. 1999. *Business @ the speed of thought*. London: Penguin Books
- Goldratt, E.M.& Cox, J. 1990. *The Haystack Syndrome*. Sifting Information out of the data Ocean.US: North River Press
- Goldratt, E.M.& Cox, J. 1992. *The Goal. A process of ongoing improvement*. Cape Town: Creda Press
- Graham, D. & Gilb, T. 1993. *Software Inspection*. London: Addison-Wesley

- Graham, D. & Goldsmith, R.F. 2002. The Forgotten phase. *Software Development*, Vol 10 (i7) part 1: p45-48.
- Graham, D. & Goldsmith, R.F. 2002. This or that, V or X? *Software Development*, Vol 10 (i8) part 2: p43-46.
- Graham, D. & Goldsmith, R.F. 2002. Proactive Testing: not V, not X. *Software Development*, Vol 10 (i9) part 3: p42-46.
- Graham, D. & Goldsmith, R.F. 2002. Test-Driven Development. *Software Development*, Vol 10 (i10) part 4 p45-47.
- Graham, D. 2002. Requirements and Testing: Seven Missing-Link Myths. *IEEE Software*. Vol 30. September/October 2002. pages 15-17
- Groenmeyer, G. 2004. Personal Communication. 28 April, Pinelands, Cape Town.
- Gulke, W. 2000. *Ten lessons from the future. 21st Century Impact on Business, Individuals and investors*. Centurion South Africa: @ One Communications
- Gulke, W. 2001. *Lessons in radical innovation*. Centurion South Africa: @ One Communications.
- Grunder, F. 2004. Personal Communication. 20 June, Goodwood, Cape Town.
- Hansen, J. I. & Thompson, C. A. 2002. Panning for Gold. Knowledge Management at work. Limra International Inc. US.
- ISO/IEC. International Standard. Information Technology – *Software packages – Quality requirements and testing*. First edition 1994. Geneva Switzerland. Reference number ISO/IEC 12119.
- Kalakota, R. & Robinson, M. 2001. *e-Business 2.0*. US: Addison-Wesley.
- Kaner, C, Falk, J. & Nguyen, H.Q. 1999. *Testing Computer Software*. Canada: Robert Ipsen.
- Kit, Edward 1997. *Software Testing in the Real World. Improving the process*. New York: Addison–Wesley.
- Kochikar, V. P. 1999. *The Knowledge Management Maturity Model – A Staged Framework for leveraging knowledge*. Infosys Technologies Limited. Bangalore India

- Koenig, M.E.D. 2002. Time saved – a misleading justification for KM: The bottom line for KM is better decision making and higher productivity. *Gale Group. Information Integrity. KM World*, Vol 11 (i5): page 22.
- Lanowitz, T. & Gassman, B. 2002. Vendor rating: Mercury Interactive. Gartner research article.
- Laudon, J. P. & Laudon, K. C. 1996. *Management information systems. Organisation and Technology*. New Jersey: Prentice-Hall
- Laudon, J. P. & Laudon, K. C. 2002. *Management information systems. Managing the Digital Firm*. New Jersey: Prentice-Hall
- La Fleur, G. 2004. Personal Communication. 19 April, Pinelands, Cape Town.
- Lewis, W. E. 2000. *Software Testing and continuous Quality Improvement*. US:. Auerbach CRC Press LLC.
- Malhotra, Y. 2001. *Knowledge Management and Business Model Innovation*. London and US: Idea Group Publishing.
- Marchand, D. A. 2000. *Hard IM choices for senior managers*. In Financial Times: Mastering Information Management. Complete MBA companion in information management / Edited by T. H. Davenport, T. Dickson & D. A. Marchand. Harlow UK: Prentice Hall
- Marick, B. 2003. *Classic Testing Mistakes*. [online].
Available: <http://www.testing.com/writings/classic/mistakes.html>
- McGregor, J. D. 2000. *The Testing Perspective*. SIGS Publication. July/August 1999a. Pages not numbered
- Monk, K. & Malone, M. 1998. *Explicit Management of the Knowledge Asset*. CSC Research Report. Sussex UK: Flexiprint Ltd.
- Mosley, D.J. 2001. *Automated Software Testing. How I fought the Software Testing Automation War and have determined I'd rather be a Stone Mason*. Publisher unknown.
- Mouton, J. 2001. *How to succeed in your Master's & Doctoral Studies. A South African Guide and Resource book*. Pretoria: Van Schaik publishers

- Murray, P. 2000. *How smarter companies get results from KM*. In Financial Times: Mastering Information Management. Complete MBA companion in information management / Edited by T. H. Davenport, T. Dickson & D. A. Marchand. Harlow UK: Prentice Hall
- Nonaka, I, Krogh, V. G. & Ichijo, K. 2000. *Enabling Knowledge Creation*. Oxford: University press
- Perry, W.E. & Rice, R.W. 1997. *Surviving the top ten challenges of software testing*. New York: Dorset House Publishing
- Pettichord, B. 2000. Testers and Developers think differently. *Software Testing & Quality Engineering*. Jan/Feb 2002
- Paul, J., Rashka, J. & Dustin, D. 1999. *Automated Software Testing: Introduction, Management and Performance*. Indianapolis:. Addison–Wesley
- Patel, N., Govindrajan, M. , Maharana, S. & Ramdas, S. 2001. *Test Case Point Analysis*. Cognizant Technology Solutions.
- Paulk, M.C. [et al] 1995. *The Capability Maturity Model: Guidelines for improving the Software Process*. Carnegie Mellon University Software Engineering Institute. US: Addison-Wesley Publishing
- Pietersen, W. 2002. *Reinventing Strategy*. New York: John Wiley & Sons
- Prusak, L. 2000. *Making knowledge visible*. In Financial Times: Mastering Information Management. Complete MBA companion in information management / Edited by T. H. Davenport, T. Dickson & D. A. Marchand. Harlow UK: Prentice Hall.
- Rodrigues, M. J. 2002. *The New Knowledge Economy in Europe*. Edward Elgar Publishing Ltd
- Rommel, G. [et al] 1996. *Quality Pays*. Kent UK: Macmillan Business
- Sage, P. 2004. Personal Communication. 8 April, Test and Data Services, Pinelands, Cape Town.
- Saunders, M., Lewis, P. & Thornhill, A. 2000. *Research Methods for Business Students*. Harlow England: Pearson Education Limited
- Schach, S. R. 1993. *Software Engineering*. Boston US:. Asken Associates Inc

- Schmaman, I. 2004. Personal Communication. 19 April, Goodwood, Cape Town.
- Senge, P. M. 1990. *The Fifth Discipline. The Art and Practice of the Learning Organisation*. New York: Doubleday
- Skyrme, D. J. 1999. *Knowledge Networking: Creating the collaborative enterprise*. Butterworth-Heinemann, Oxford: Reed Educational and Professional Publishing.
- Snowden, D. & Andrews, P. 2002. *Next Generation Knowledge Management: The Complexity of Humans*. IBM. Executive Trek Report (18 November 2002).
- Snowden, D. 2003. *Tacit and Explicit Knowledge: an organic approach to knowledge strategy and knowledge mapping*. [online].
Available: http://www.kmeurope.com/ds_msclass.asp
- Stacey, R.D. 2001. *Complex Responsive Processes in Organisations*. MPG Books Ltd
- Stewart, T.A. 1997. *Intellectual Capital - The new wealth of organisation*. New York: Nicholas Brealey Publishing Ltd.
- Sullivan, P. H. 2000. *Value-driven Intellectual Capital. How to convert intangible corporate assets into market value*. Canada: John Wiley & Sons
- Sveiby, K. E. 1997. *The new organisational wealth. Managing and measuring knowledge-based assets*. San Francisco US: Berrett-Koehler Publishers
- Sveiby, K. 2001. *What is knowledge management?* [online]
Available: <http://www.sveiby.com/articles/KnowledgeManagement.html>
- Swartz, N. 2003. *The wonder years of Knowledge Management*. Ebsco Publishing
- Swartz, J. 2004. Personal Communication. 26 April, Compuware V&A Waterfront, Cape Town.
- Swieringa, J. & Wierdsma, A. 1992. *Becoming a Learning Organisation. Beyond the learning curve*. Cambridge UK: University Press
- Tapscott, D. & Caston, A. 1993. *Paradigm Shift. The new promise of Technology*. Baskerville US: McGraw-Hill, Inc.
- Thomas, D. & Hunt, A. 2001. *Learning to love unit testing*. [online]
Available: <http://www.stqemagazine.com/featured.asp>

- Whitten, J. L.(2001). *Systems Analysis and Design Methods*. New York: Irwin McGraw-Hill publishers
- Wiegers, K. 1997. Seven Deadly Sins of Software Reviews. *Software Development*. Vol 7. March 1997
- Yaquinta, J. & Blaney. date unknown. *The Virtual Test Team*. Gartner Research article.
- Zack, Michael H. 1999. *Knowledge and Strategy*. US: Butterworth-Heinemann, Reed Elsevier Group.
- Zack, M. H. 1999. *Developing a Knowledge Strategy*. California Management Review. Vol 41. Number 3 1999. pp 125 –143.

APPENDIX A

SOFTWARE MODELS

There are several existing life cycle process models, such as the Waterfall model, the V-model, various Spiral Models, Gelperin and Hetzel's Evolutionary Testing Model, Beizer's Progressive Phases of Tester's Mental Model to mention a few.

Burnstein et al says that we must recognize that software testing is pervasive and as a result it influences the test models and thereby the test methods used in organisations by individuals that interpret, analyse and apply it to the situation at hand, or apply it according to experience or knowledge disposition. By pervasive it is meant that testing does not start and stop by strict adherence to clinical time-frames as the traditional waterfall method would suggest in the Systems Development Life Cycle (SDLC), (see Figure 14). The waterfall approach breaks the development cycle down into discrete phases each with a rigid sequential beginning and ending.

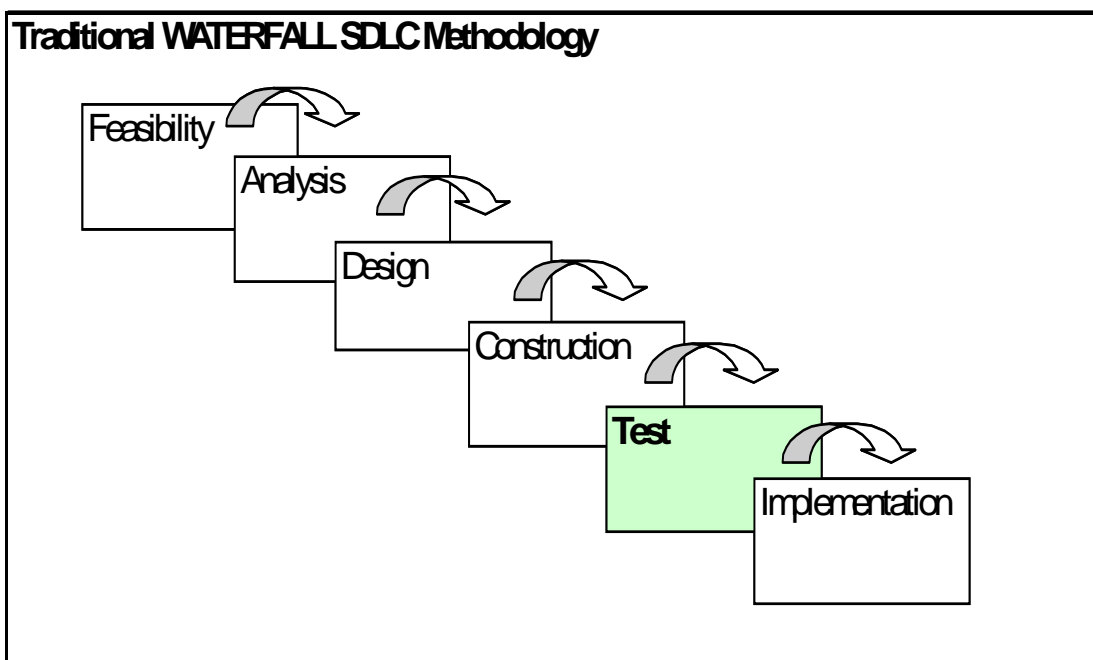


Figure 14: Appendix A – Traditional Waterfall SDLC Methodology

The waterfall model suggests that each phase should be fully completed before the next is started. And once a phase is completed, in theory one never goes back to it.

There are a number of other approaches as well, such as the Spiral Test approach and the Deming PDCA (Plan, Do, Check, Act) models.

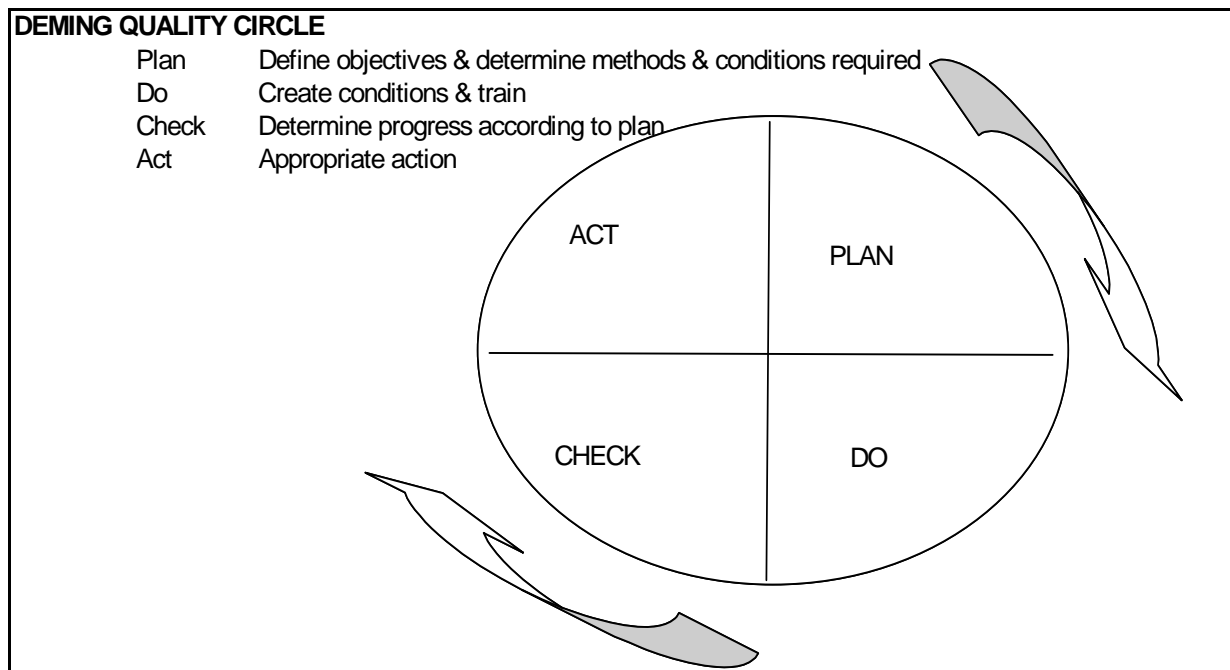
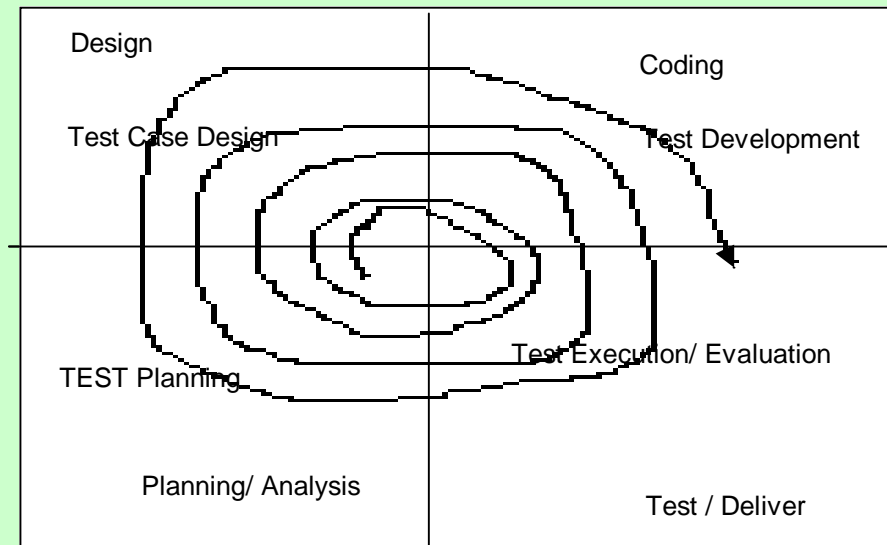


Figure 15: Appendix A – Deming’s PDCA quality circle

The Spiral models are a direct reaction to the traditional waterfall methodology. A problem with the waterfall model is the elapsed time for delivering the product may be excessive. By contrast, spiral models expedite product delivery. Discrete functions span build-test iterations at an accelerated rate. One advantage is that clients receive functionality quickly. Another is that the delivery can be shaped by continuous rapid feedback (see Figure 16). A disadvantage in an immature culture is the tension that the erratic rapidity creates between testers continuously logging bugs (software errors) and developers resolving these whilst under pressure to complete and hand-over delivery. The delivery also runs the risk of never completing, as its status does not change from constantly being ‘delivery under test’ – also known as a spiral of death.

Spiral Development/ Test Methodology

Opposed to traditional, users involvement throughout
Product shaped iteratively



Expedites product delivery
Risk of spiral death ...no end

Figure 16: Appendix A – Spiral Development / Test Methodology

APPENDIX B

BUG CYCLE

The aim of reporting problems is to bring them to the attention of appropriate people, who will assess, prioritise and schedule these bugs to be fixed. The test team will either confirm or rebut these fixes. It stems from the last step in the Test cycle when a deviation of requirement or failure occurs.

The process flow of a BUG cycle is:

- **Assess.** When a tester logs an error it is first assessed for duplication or validity for that project. For example, for a release project an error of the previous release will not be accommodated as a bona fide error of the new release. Only current release errors will qualify for the release project under test.
- **Rejected.** If the reviewer assesses the log and has insufficient detail, or considers it a duplicate, or requires clarity. The reviewer will inform the tester or the Test manager / leader.

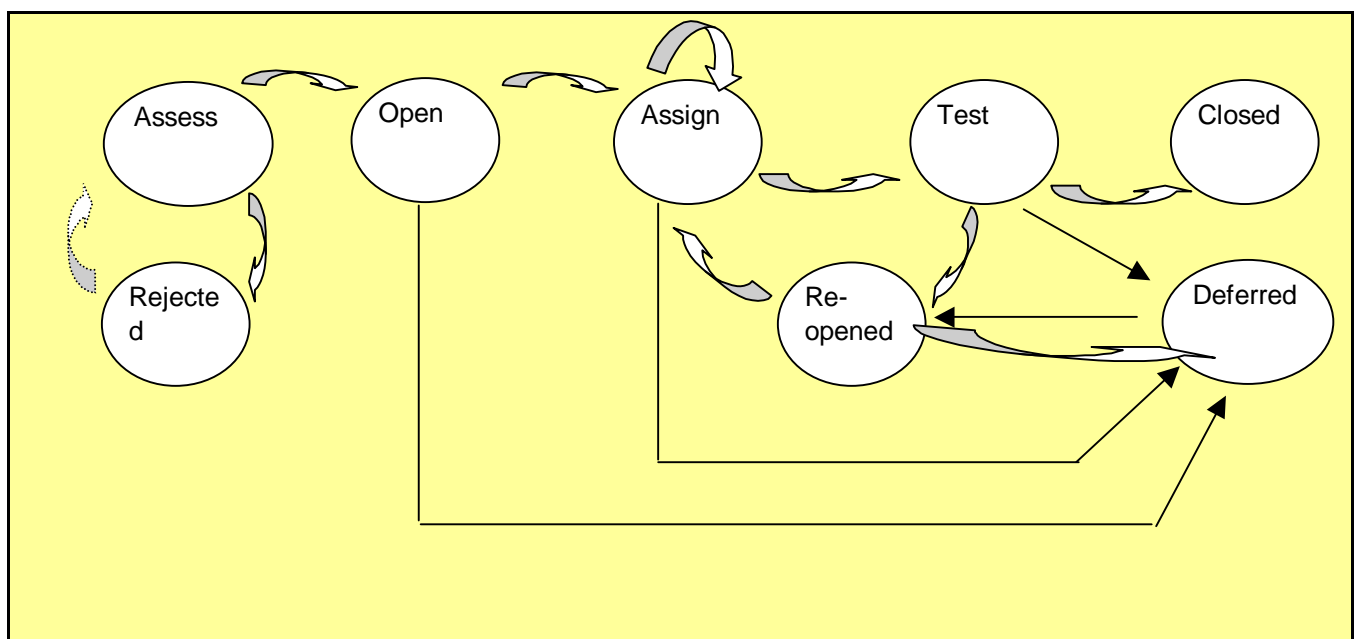


Figure 17: Appendix B – Bug Cycle (Source: Black, 2002)

- **Open.** After assessment, a unique error number is attached.
- **Assigned.** The error is assigned to the appropriate resource. For errors that cannot be resolved internally, these will be logged with the vendor.
- **Test.** On response from the reviewer, or a member of the configuration team, or IT indicating that the error has been fixed, the tester regression tests to confirm resolution and to ensure that no new errors have been found as a result of this fix.
- **Reopened.** If the fix fails, the tester reopens the error.
- **Deferred.** If under time or other constraints, and the error is a low severity, low priority, the error could be deferred to be tested at a more appropriate timeslot or consolidated into another project associated with that discrete functionality.
- **Closed.** If the fix passes regression testing, the tester closes the bug.

A good bug-tracking tool should enable the gathering and management of the information.

APPENDIX C

EMPLOYEE BENEFITS - AN OVERVIEW AND ARCHITECTURAL COMPONENTS

Never before has there been such enormous pools of money held in developed countries by institutional investors, primarily in pension funds. In the US alone it is estimated that an average pension funds holds assets of \$80 billion; even a small pension has more than \$1 billion invested in the economy (Drucker, 1993). These pools of capital dwarf anything the greatest 'capitalist' commanded. The age structure, technology and the global network of a developed or developing society virtually guarantees that pension funds will become profoundly important in every single country (Drucker, 1993). Pension Funds, also known as Employee Benefit Funds, is significant, and an intimate part of the fabric of society.

Many rules regulate EB Funds, many institutions keep a watchful eye over it, and many employees depend on it, many Financial Service Providers profit from it. It is complex in its architectural structure, internal functions and actuarial computations. To be truly regarded as a credible authority takes years of experience in the field. This experience is complemented by a period of study (for e.g. the completion of an exam to earn worthy membership to the Institute of Life and Pensions Advisers (ILPA)).

This chapter gives a high-level macro view of EB and its components with the primary aim of showing the degree of domain knowledge required to achieve effective software testing.

To gain a better understanding we start with EB in lay terms, move on to the constitution of core benefits and then for a better understanding, provide a high level schematic of EB architecture.

EMPLOYEE BENEFITS IN LAY TERMS

There are two obvious but distinct parts to the statement Employee Benefits (EB). (a) employee and (b) benefits

An EB package reflects the needs, desires and *benefits* of *employees* as perceived by an employer (or a Union or an Industrial Council) and influenced by his budgetary cost constraints. The employer carefully considers the correct construction of the benefits package to suit the employees through the vehicle of a Pension Fund.

A benefits programme is not a single once off creation. It is reviewed regularly to ensure it matches and continues to answer the changing conditions of the economy, the choices employees' desire and offers the best yield that will be paid on retirement, death, disability, ill health, or withdrawal.

BENEFIT STRUCTURE

This is the rub. The complications, combinations and complexity of EB options are many. The majority of these complexities service the multi-faceted and increasing needs of employees. A certain amount of complexity also enters the EB industry by various stakeholders such as SARS (South African Revenue Services), The Pensions Fund Registrar, Trustees, Brokers and Agents, Life Office Association, Actuaries, Financial Services Board, Underwriters, employers and not forgetting the employees themselves.

There are various types of Funds that are recognized and defined by the Registrar of Pension funds in the Pension Funds Act, the terminology that may differ at times from the Income Tax Act and the Commissioner of Inland Revenue (ILPA, 86).

Being a multi billion currency industry it is under scrutiny of watchdogs, ombudsman and various Acts that regulate, govern, enable and/or limit operators in the field.

With flexible choice driven by technology, a typical company or institute's Employee Benefit's package could offer employees any number of benefits.

Employees being human, and their desires subject to change base their commitment on personal circumstances, social influences, economic conditions, legal status, and global or local political reform in order to choose a benefit structure that suits them at a point in time.

In the main, the benefits which make up the core of EB products are analysed and impacted by the following variables (Table 15):

Preconditions of benefits entitlement	Calculation factors	Beneficiaries	A select bunch of Benefits
Minimum term of service	Salary	Member	<ul style="list-style-type: none"> • Normal retirement benefits • Lump sums • Early Retirement • Late retirement • Death Benefits • Severance Benefits • Retrenchment Benefits • Discontinuance benefits
Age	Term of service	Dependants (spouse, orphans, widow.....etc).	
Category of employee	Fixed amounts	Estate	
Life Expectancy	Aggregate of past salary	Spouse	

Table 15: Appendix C - EB Benefits and variables

Service providers with the help of expert actuaries, and competent professionals with years of experience in the field of Employee Benefits compute differentiating products that they present and distribute to discretionary consumers.

There are organisations that offer the capability of servicing every combination of needs, and these lead to complex administrative solutions. These intricate solutions that deal with huge amounts of money and sensitive information are impossible to be supported on a manual basis - they are supported by experienced knowledge workers and by complex computer systems.

The Employee Benefits administrative systems architectures although tremendously crowded with information are best illustrated diagrammatically. The diagram (Figure 18) imparts a semblance of understanding this vast application.

EMPLOYEE BENEFITS ARCHITECTURAL COMPONENTS

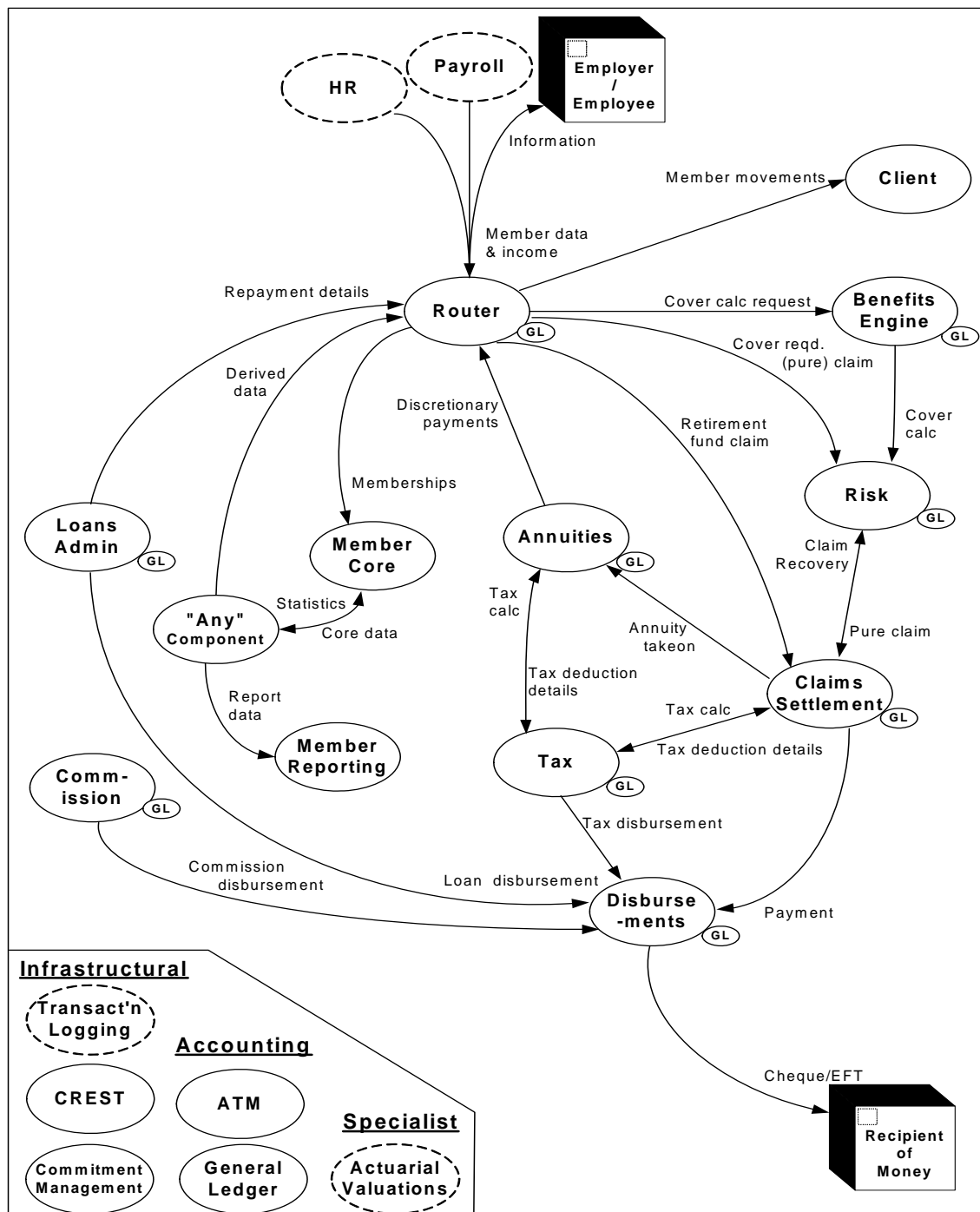


Figure 18: Appendix C – EB Architectural Components (Source: Schmaman, 2003)

SUMMARY

The above is a synoptic view of EB - by no means exhaustive. It is not the intention of this Appendix to provide a functional picture of Employee Benefits, but to present typically the compounded complexity of an evolving consumer-centric application and to show what impact it has. In essence, to show the importance of domain knowledge, and how it impacts and influences Software Testing.

APPENDIX D

COVERING LETTER

RESEARCH SURVEY: A KNOWLEDGE APPROACH TO SOFTWARE TESTING IN AN EMPLOYEE BENEFITS ENVIRONMENT

I am a part-time Masters student at the University of Stellenbosch in the department of Information Science and Knowledge Management and I am currently conducting research into the value and influence of knowledge in the testing phase of systems development primarily in an Employee Benefits environment.

‘Knowledge has become the key economic resource and the dominant – and perhaps the only – source of comparative advantage’ – Peter Drucker.

The Employee Benefits market (or Retirement Fund market as it is also known) is an expansive and extremely competitive market. Peter Drucker, one of the well-known KM contributors in his book 'Post Capitalist Society' expresses the extent, depth and importance of Retirement Fund markets in a country's Gross National Product. He states that EB investments and products contribute more than 75% of the money-flow in the United States. EB is significant, cannot be ignored and is with us now and will certainly be with us in the future.

It is worth investigating the extent, influence and value of a knowledge approach in the test phase of the systems development life cycle of an EB application to verify if ‘one-size-fits-all’.

Defect free implementation of software is a desired objective. The frequency of software and configuration changes to accommodate different or customised (EB) markets makes defect free software implementation at best complex, and at worst an elusive goal. Management budget a great deal of money, attention, and a significant amount of resources to attain a trouble-free resilient delivery to market. And efficient and effective software testing is vital component of such a quality delivery

The variables that influence the quality of delivery in Employee Benefits are many, and much of it is people-centric including:

- the experience or maturity level of testing (more specifically the testers (test) capability)
- conformance to a test methodology
- embedded technical or content knowledge of the application
- domain or business knowledge
- the culture of sharing knowledge in the team
- leadership
- organisational structure
- knowledge of support processes
- understanding of the business and accurate transformation/articulation of this into proper source documents

Given your profile, the experience that you hold will add value to this research. In anticipation I appreciate your time and effort taken to complete this questionnaire. The questionnaire should take no more than 30 minutes to complete and may be returned to me via email or printed and mailed to me (refer address on the last page of the questionnaire – I shall defray any costs incurred).

All information provided in this survey will be treated in the strictest confidence and no individual details will be published in any of the reports. Please indicate in completing the questionnaire if you would like to receive a copy of the final report.

Should you have any queries regarding this survey please contact me on 082 565 9368 or via email at essackm@xsinet.co.za. I thank you for your participation. I am confident that the results of this survey will be of value to Software Testing community and in particular, the EB environment.

Yours sincerely
Essack Mohamed
Test Manager

APPENDIX E

SURVEY INSTRUMENT

A KNOWLEDGE APPROACH TO SOFTWARE TESTING IN AN EMPLOYEE BENEFITS ENVIRONMENT

Please answer all questions as accurately as possible in the context of software testing.

There are no right or wrong answers. This questionnaire is intended to:

- extract elements of software testing (in an EB environment)
- capture your opinion to the questions asked, and
- provide an assessment of software testing based largely on people-centric criteria.

Important Definitions

For the purposes of this questionnaire **knowledge sharing** should be seen as **any** exchange that results in knowledge transfer from one person to another. This may range from formal feedback sessions or meetings to corridor discussions, tea-pause chats or email questions and answers.

Knowledge approach should be seen in the context of knowledge sharing and may include the use of documents, templates, models, patterns or tools.

Please answer the following questions by completing key words or placing an X in the appropriate box. If a question is not applicable please mark N/A.

1.State your current role / job title

2. Are you ☐ Permanently employed or ☐ A contractor?

3. How old are you?

4. Gender ☐ Male or ☐ Female

5. How many years total work experience do you have?

6. How many years experience do you have in information systems / technology?

7. How many years experience do you have in Employee Benefits?

8. How many years experience do you have in software testing?

9. Do you have a tertiary qualification? ☐ Yes ☐ No

Specify:

Institution

Qualification

10. Do you have formal Software Testing training? ☐ Yes ☐ No

Specify type

Institution

Duration

11. Select the organisational size that best describes your current company? (Your employer, not the company you may be consulting at)

- ☐ < 30 employees ☐ 31 – 100 employees ☐ 101 - 250 employees
☐ >250 employees

12. List the types of documents/instruments that you usually use or prepare in your organisation related to Software Testing.

- ☐ Test Strategy
- ☐ Test Approach
- ☐ Test Plans
 - ☐ Generic, or
 - ☐ Specific per test phase, or
 - ☐ Both
- ☐ Test Decision Tree
- ☐ Requirements Matrix
- ☐ FMEA (Failure Mode and Effect Analysis) spreadsheet
- ☐ Risk Assessment profile
- ☐ Test Priority ranking
- ☐ Test schedule
- ☐ Test Summary reports
- ☐ Application Readiness reports
- ☐ Checklists
- ☐ Other
- ☐ Specify:

		Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
13	I follow a formal methodology in Software Testing. Specify					
14	I regularly use Software Testing tools / software. Specify which How do these tools support knowledge sharing or the quality of the delivery?					
15	I regularly use templates. Specify which					
16	Are you familiar with the concept of a Test Life Cycle? <input type="checkbox"/> Yes <input type="checkbox"/> No Please specify					
17	I regularly reuse existing knowledge when performing my test function. Specify circumstances.					
19	Knowledge sharing is a high priority with respect to testing in my organisation. If applicable, describe briefly how knowledge sharing happens in your organisation.					

		Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
20	I regularly maintain a personal knowledge base. If so, state format (paper, excel / word etc.) And content					
21	I contribute to a corporate knowledge base. If so, how often?					
22	When I am faced with a new project in an area I have not worked in before, I will first use a knowledge base or draw on the knowledge of others in this area before starting.					
23	In an attempt to reduce my test risk, I use a knowledge base or draw on the knowledge of others in this area.					
24	The culture of my current organisation does not encourage knowledge sharing.					

25. Which areas do you believe are valuable to collect and share data on within software testing?

Is this currently done in your company?

If so, what value does it add?

26. Do you think that using a knowledge sharing approach is valuable in software testing?

☐ Yes

☐ No

Why?

27. **Rank** the following items, in order of importance, as they are likely to influence your quality of delivery

1 = Most Important 2 = Fairly important
3 = Medium importance 4 = Least important

- _____ Application Domain knowledge (for e.g. the depth or extent of EB knowledge)
- _____ Software Testing competency and knowledge
- _____ Credibility of source (and reliability on source documents)
- _____ Timely involvement/engagement of testing in the SDLC
- _____ Maturity level of testing and flexibility to adapt existing knowledge

28. How would you rate the relative importance of the following benefits of knowledge sharing in the context of EB and in the context of your software testing experience?

1 = critical; 2 = very important; 3 = important;
4 = somewhat important 5 = irrelevant

- _____ Saves time in finding a solution
- _____ Gives a better quality solution
- _____ Synergistic value of many contributors to the solution
- _____ Decreased cost due to avoiding the “same mistakes”

29. How would you rate the following perceived drawbacks to sharing knowledge?

1 = critical; 2 = very important; 3 = important;
4 = somewhat important 5 = not really relevant

- _____ Too time consuming
- _____ Cost of implementing and maintaining is too high
- _____ Tools are inadequate to support knowledge sharing
- _____ Do not get credit for sharing knowledge
- _____ Intellectual ownership (loss of “power” through knowledge sharing)

30. How would you rate your personal attitude to knowledge sharing? (Tick the **most appropriate** box)

- ☐ I will actively look for opportunities to share my knowledge and that of others
- ☐ If opportunities arise that require me to share my knowledge I will do so willingly
- ☐ Knowledge sharing is a good idea but I don't find time to do it
- ☐ I tend to oppose knowledge sharing
- ☐ I am strongly opposed to knowledge sharing

31. How would you rate the importance of the following elements in its ability to influence/impact the software testing process or the quality of the delivery (in an EB environment).

1 = critical; 2 = very important; 3 = important;
4 = somewhat important 5 = not really relevant

_____ Lack of EB business knowledge

_____ Lack of testing knowledge

_____ Lack of application knowledge

_____ Lack of access to EB knowledge

_____ Lack of access to proper testing knowledge

_____ Lack of access to application knowledge

Please rate the adequacy of the following elements as a current snap-shot of software testing in your organisation. Mark the appropriate box with an 'X' or N/A if not applicable.

No.	Criteria	Very adequate	Adequate	Inadequate	None
32.	Adequacy of formal training in software testing that you have received in the practice of planning, test case development, test execution, results analysis and reporting.				
33.	Adequacy of equal sharing of success and failure across the team (considered a single team in the delivery)				
34.	Adequacy of the tester's inclusion in the entire SDLC; specifically that the testers are aware of the objectives and high level requirements.				
35.	Adequacy of the tool sets available to testers before execution.				
36.	Adequacy of training and support testers receive in using the set of tools.				

37.	Adequacy of management's time and attention to testing, as well personal interest in the challenges and problems associated with testing.				
38.	Adequacy of management's commitment and investment in the testing process.				
39.	Adequacy of the profiles that testers have of EB business especially concerning the specific needs and complexities of EB.				
40.	Adequacy of the testers' interaction with all levels of EB users, particularly of the feedback and support received from them.				
41.	Adequacy of the test management involvement in adjusting testing schedules and budgets as they are impacted by slippages and scope changes.				
42.	Adequacy of communication and of communication opportunities between developers/configuration team and testers to plan for and discuss test results.				
43.	Adequacy of response times, acceptance and resolution from developers or configuration team members to the test results/failures				
44.	Adequacy of the definition of the requirements and of the design for developing test conditions that will enable the testers to validate entrance or completion criteria				
45.	Adequacy of the documentation of changes to the initial requirements and design so that testers can modify test criteria to validate the current software requirements and design criteria.				
46.	Adequacy of the shared understanding across the stakeholders as to the quality, the acceptable conditions of satisfaction and the reliability levels expected.				

47. On average, given your current software testing status, do you experience more software implementation failures or more successes?

For the purposes of this questionnaire, a failure is regarded as:

- Omitting, missing or not meeting all the business requirements, or
- Not satisfying all of the completion or exit criteria, or
- Implementing despite high risk (deployment with open severity 1 or 2 bugs), or
- Not attaining sign-off, or
- Deployment failure and/or recovery required (including partial or full recovery directly related to the deployment)

More failures

☐

More successes

☐

I would like to receive a copy of the final report.



Please provide either:

a) an email address:

OR

b) a full postal address:

My sincere appreciation - thank you for your participation.

Please return to:

essackm@xsinet.co.za or

Essack Mohamed

17 Suikerbos Street

Durbanville

7550

APPENDIX F

EARLY TESTING IS COST EFFECTIVE TESTING

PREVENTION IS CHEAPER THAN CURE.

It is clear that the single largest component of software cost is the cost of bugs: the cost of detecting them, the cost of correcting them, the cost of designing test that discover them, and the cost of running those tests. Consequently, the aim of software quality assurance should be bug prevention (Beizer, 1984).

Error Removal Cost Multiplies Over SDLC

SDLC PHASE	Cost
Definition	\$1
High Level design	\$2
Low level design	\$5
Code	\$10
Unit Testing	\$15
Integration Testing	\$22
System testing	\$50
Post Delivery	\$100+

Table 16: Appendix F - Early testing: Cost of error removal (Source: Paul; Automated Software Testing).

The earlier in the life cycle that errors are uncovered, the easier and less costly they are to fix. Cost is measured in terms of time and resources required to correct the defect.

A defect found earlier in the life cycle has less operational impact and usually further from integration or interfaces with other systems or units of delivery. In contrast, a defect discovered during the operational phase can involve several interfaces and perhaps even organisations - this will require a wider range of retesting and may cause downtime. Table 16

outlines potential cost savings of error detection through the various stages of the systems development life cycle.

APPENDIX G

GLOSSARY OF TERMS

Acceptance Testing, User Acceptance Testing (UAT) – A testing phase designed to demonstrate that the system under test meets business requirements. The name ‘user acceptance testing’ can imply that users run the testing

Application Domain skills (subject matter expertise) – Skill or understanding pertaining to whatever field or task being dealt with. How the business, customers or users skillfully employ the application and in depth knowledge of this sector.

AUT – Application Under Test. A complete coverage deals with the hardware, software and infrastructure being tested.

Black Box Test - “Black box” or “functional” testing tests the external behaviour or functionality of a unit without considering its internal structure. Tests based on what externally observable things a system should do. Spelt out in requirements and high level design specifications.

Bug, Defect. – A problem that causes or would cause the system to fail to meet one or more of the user’s or customers requirements or expectations of quality

Completion Criteria - The standard by which a test objective is measured, which is unambiguous and clear about when a test has succeeded.

Regression Testing – Testing to ensure what worked / functioned prior to a change continues to function in the same manner.

SDLC – Systems Development Life Cycle

Smoke Test or Sanity Test – A test run against a proposed test release to ensure that it is stable enough to enter testing in the currently active test phase. It is usually a subset of the overall set of tests, that touches every part of the system in a cursory way. A good smoke test at times runs long enough to show problems arising with reliability and availability.

System Test - Verifies that the system meets both its functional and non-functional requirements

Test Analyst - A role needed to develop test plans and also execute tests. Will provide consulting services during logical or conceptual design, technical specifications, functional specifications and developing Unit, Integration and Systems Test Plans

Test Case - A unique or specific 'pack' of test data along with expected results for a particular test objective. A sequence of steps consisting of actions to be performed on the application under test (these steps are sometimes called the test script). These actions are often associated with some set of test data (preloaded or input during the test).

Test Coverage –

1. Structural: the extent to which the test covers, or exercises, the structure – the code, subsystems, or components – the application under test.
2. Behavioural: the extent to which the test system covers, or exercises the behaviour – risks to quality, operations, activities, functions, and other uses – of the application under test.
3. Readiness: the extent to which the test system covers, or exercises the readiness – risks to quality, performance, security, regression, stress/volume where applicable, business cycles and live simulation – of the application under test

Test Data - Input data and file conditions associated with a particular test case

Test environment - includes the hardware, software, networking and other infrastructure, paper and other supplies, facilities, lab, and so forth that the test team procures, installs, configures to simulate the host system under test.

Testing processes - include both the written and unwritten procedures, checklists, and other agreements about the way the test team does its testing, in brief a test methodology.

Test Script - A test script is a set of instructions usually in series of strict steps for carrying out test cases, and for recording test results

Test system – The test environment, the testware, and the test execution process that the test team will use to assess the quality of the application under test.

Testware - includes tools, scripts, data, test cases, logging and tracking mechanisms, and so forth that the test team uses to do it's testing.

V-Model - The accepted model for testing software, which emphasises that testing is performed "in parallel" to the phases of the SDLC. A variation of the waterfall model that shows the planning and verification tasks down the left side of the V and the validation, testing and implementation tasks up the right side of the V. Across the V are implicit sequences that show the outputs of each phase that drive or act as source of each testing phase.

- Acceptance testing is driven primarily by requirements.
- System testing is driven by requirements and design
- Integration testing is driven by requirements, design, architectural documents, and code interfaces, and
- Unit testing is driven by requirements, design, architectural documents, code interfaces and technical specifications

White Box Test or Structural Tests - "White box" or "structural" testing organises tests based on knowledge of the unit's internal logic structure.